NASA-CR-192863

CSDL-C-5709

ADVANCED INFORMATION PROCESSING SYSTEM (AIPS)

SYSTEM SPECIFICATION (Revision 1)

October 1984

Prepared for the
Lyndon B. Johnson Space Center
of the
National Aeronautics and Space Administration
under
Contract NAS9-16023
Task Orders No. 35 and No. 84-18

[illegible]

The Charles Stark Draper Laboratory, Inc.
Cambridge, Massachusetts 02139

[illegible]

CSDL-C-5709

ADVANCED INFORMATION PROCESSING SYSTEM

(AIPS)

SYSTEM SPECIFICATION

(Revision 1)

October, 1984

Approved: [illegible]
Robert N. O'Donnell, Technical Manager

Approved: [illegible]
Philip G. Felleman, Program Manager

The Charles Stark Draper Laboratory, Inc.
Cambridge, Massachusetts 02139

ACKNOWLEDGEMENT

This report was prepared by The Charles Stark Draper Laboratory, Inc. under Contract NAS9-16023, Task Orders #35 and #84-18, with the Lyndon B. Johnson Space Center of the National Aeronautics & Space Administration.

A preliminary version of this specification was issued on March 16th, 1984 under Task Order #35, for the purpose of supporting the AIPS Phase I System Design Review at the Draper Laboratory on March 28th - 30th, 1984. Revisions incorporated after March 1984 to reflect results of the Phase I System Design Review process were accomplished under Task Order #84-18 and the document was published as CSDL-C-5709 on May 15th, 1984.

This Revision 1 of CSDL-C-5709 differs from the original (5/15/84) version in that a more extensive Table of Contents is provided. The technical content is unchanged.

Publication of this report does not constitute approval by the NASA/JSC of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

ABSTRACT

The Advanced Information Processing System (AIPS) is designed to provide a fault tolerant and damage tolerant data processing architecture that meets aeronautical and space vehicle application requirements. Quantitative and qualitative AIPS requirements derived from seven different applications have been defined. Examples of the former are processor throughput, memory size, transport lag, mission success probability, etc. Examples of the latter are graceful degradation, growth and change tolerance, integrability, etc. The AIPS architecture will satisfy the quantitative requirements and also have attributes that make it responsive to the qualitative requirements.

The system is comprised of hardware 'building blocks' which are fault tolerant processing elements, a fault and damage tolerant intercomputer network and an input/output network, and a fault tolerant power distribution system. A network operating system integrates these elements into a coherent system.

The AIPS architecture permits application designers to select an appropriate set of the building blocks and system services and configure a specific processing system for their application. The application designer need not include all the building blocks that are available in the AIPS system. The number and type of building blocks and their configuration will be determined by the specific applications requirements.

This specification defines the AIPS and specifies the hardware and software configurations for a laboratory proof-of-concept (POC) system which will be built and evaluated to demonstrate feasibility of the AIPS concept.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1.0 SCOPE

1.1 Purpose This specification sets forth the requirements for the design, development, fabrication, quality assurance and evaluation of the Advanced Information Processing System (AIPS) Proof-of-Concept (POC) System hereinafter referred to as the POC System. These requirements shall be the basis for the preparation of more detailed requirements to be included in:

(1) subsequent revisions of the system specification, and in related system documents, such as interface control documents (ICDs).

(2) specifications for the identified configuration items, hardware and software, at lower levels of assembly.

2.0 APPLICABLE DOCUMENTS

2.1 Governmental Documents None

2.2 Nongovernmental Documents

(1) Requirements

(a) AIPS System Requirements, CSDL Report No. AIPS-83-50, August 30, 1983

(2) Other Publications

(a) Software Development Policies & Guidelines, CSDL Report No. CSDL-C-5526, April 15, 1983
(b) AIPS Program Plan - Phase I, CSDL Report No. AIPS-83-38, August 24, 1983
(c) AIPS Technology Survey Report, CSDL Report No. CSDL-C-5691, February 1984

3.0 REQUIREMENTS

3.1 System Definition

3.1.1 Introduction The Advanced Information Processing System (AIPS) is designed to provide a fault tolerant and damage tolerant data processing architecture that meets aeronautical and space vehicle application requirements. The requirements for seven different applications are described in the 'AIPS System Requirements'. The requirements can be divided into two categories: quantitative and qualitative. Examples of the former are processor throughput, memory size, transport lag, mission success probability, etc. Examples of the latter are graceful degradation, growth and change tolerance, integrability, etc. The AIPS architecture is intended to satisfy the quantitative requirements and also have attributes that make it responsive to the qualitative requirements.

The system is comprised of hardware 'building blocks' which are fault tolerant processing elements, a fault and damage tolerant intercomputer network and an input/output network, and a fault tolerant power distribution

PRECEDING PAGE BLANK NOT FILMED

system. A network operating system ties these elements together in a coherent system.

The system is managed by a Global Computer that allocates functions to individual processing sites, performs system level redundancy management and reconfiguration, and maintains knowledge of the system state for distribution to the component elements. Redundancy management, task scheduling, and other local services at individual processing sites are handled by local operating systems. The network operating system links local operating systems together for such functions as intertask communications.

The following sections define the AIPS in greater detail. Section 3.1.2 is a conceptual definition of the system and its operational philosophy. Section 3.1.3 describes the system services provided by the AIPS. Section 3.1.4 is a more specific description of the proof-of-concept system configuration, the AIPS building blocks, and their architecture. The laboratory operating environment for the POC system is described in Section 3.1.5.

The AIPS architecture permits application designers to select an appropriate set of the building blocks and system services and configure a specific processing system for their application. The application designer need not include all the building blocks that are present in the POC system. The number and type of building blocks and their configuration will be determined by the specific applications requirements.

3.1.2 AIPS Concepts

3.1.2.1 Overview The Advanced Information Processing System consists of a number of computers located at processing sites which may be physically dispersed throughout the vehicle. These processing sites are linked together by a reliable and damage tolerant data communication bus, called the Intercomputer Bus or the IC bus.

A computer at a given processing site may have access to varying numbers and types of Input/Output (I/O) buses. The I/O buses may be global, regional or local in nature. Input/Output devices on the global I/O bus are available to all or at least a majority of the AIPS computers. Regional buses connect I/O devices in a given region to the processing sites located in their vicinity. Local buses connect a computer to the I/O devices dedicated to that computer. Additionally, there may be memory mapped I/O devices that can be accessed directly by a computer similar to a memory location. Finally, there is a system Mass Memory that is directly accessible from all computers in the system on a dedicated multiplex Mass Memory (MM) bus.

Input/Output devices on the global I/O bus are available system wide. The global I/O bus is a time division multiple access (TDMA) contention bus. The intercomputer (IC) bus is used to transmit commands and data between computers. The IC bus is also a TDMA contention bus.

Figure 1 on page 4 shows a simplified system level diagram of the AIPS architecture.

Computers at various AIPS processing sites are General Purpose Computers (GPCs) of varying capabilities in terms of processing throughput, memory, reliability, fault tolerance, and damage tolerance. Throughput may range from that of a single microprocessor to that of a large multiprocessor. Memory size will be determined by application requirements. Reliability, as measured by probability of failure due to random faults, ranges from 10^{-4} per hour for a simplex processor to 10^{-10} per hour for a multiprocessor that uses parallel hybrid redundancy. For those functions requiring fault masking a triplex level of redundancy is provided. For low criticality functions or noncritical functions, the GPCs may be duplex or simplex. Parallel hybrid redundancy is used for extremely high levels of fault tolerance and/or for longevity (long mission durations). GPCs can also be made damage tolerant by physically dispersing redundant GPC elements and providing secure and damage tolerant communications between these elements. Within AIPS, computers of varying levels of fault tolerance can coexist such that less reliable computers are not a detriment to higher reliability computers.

The overall framework in which AIPS operates can be characterized as a limited form of a fully distributed multicomputer system. A fully distributed system must satisfy several requirements. The following subsections describe these requirements and characterize the AIPS architecture in the context of these requirements.

3.1.2.2 Function Migration A fully distributed system must have a multiplicity of resources which are freely assignable to functions on a short-term basis. AIPS has multiple processing sites; however, they are not freely assigned to functions on a short-term basis. During routine operations the General Purpose Computers at various processing sites are assigned to perform a fixed set of functions, each computer doing a unique set of tasks. However, in response to some internal or external stimulus, the computers can be reassigned to a different set of functions. This results in some functions migrating from one processing site to another site in the system. Under certain conditions, it may also result in some functions being suspended entirely for a brief time period or for the remainder of the mission. In AIPS this form of limited distributed processing is called semi-dynamic function migration.

The internal stimuli that result in function migration may consist of detection of a fault in the system, a change in the system load due to a change in mission phase, etc. An example of an external stimulus is a crew initiated reconfiguration of the system.

3.1.2.3 Resource Transparency Another characteristic of a fully distributed system is that the multiplicity of resources should be transparent to the user. To a large extent, this is true in the AIPS. Function migration is transparent to the function and the person implementing that function in software. Interfunction communication is handled by the operating system such that the location of the two communicating functions is also transparent to both. The two functions could be collocated in a GPC or they may be executing in different GPCs. Indeed, at one time they may be collocated, while at a later time one of them may have been migrated to another site. This transparency is achieved through a layered approach to interfunction communication. One of these layers determines the current

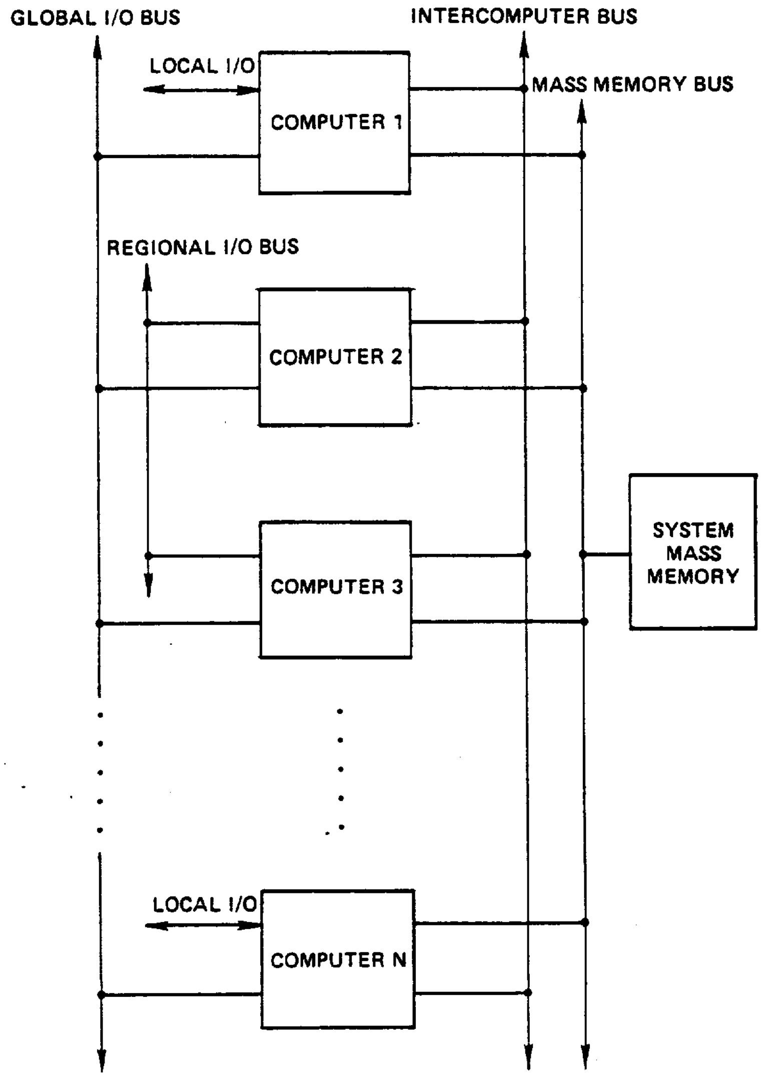

Figure 1. AIPS Architecture: A Software View

processing site of the function to which one wishes to communicate. If it is another GPC, another layer in the communication hierarchy is invoked that takes care of appropriate IC bus message formatting and interface to the bus transmitters and receivers, that is, the physical layer. This layered approach is responsible for hiding the existence of multiple computers from the applications programmer.

3.1.2.4 System Control Another characteristic of a totally distributed system is that the system control is through multiple cooperating autonomous operating systems. The AIPS operational philosophy differs considerably in this regard. The overall AIPS system management and control authority is vested in one GPC at any given time. This GPC is called the Global Computer. All other GPCs are subservient to this GPC as far as system level functions are concerned. However, all the local functions are handled quite independently by each computer. This philosophy is more akin to a hybrid of hierarchical and federated systems. This is explained in the following.

Under normal circumstances each GPC operates fairly autonomously of other computers. Each GPC has a Local Operating System that performs all the functions necessary to keep that processing site operating in the desired fashion. The local operating system is responsible for an orderly start and initialization of the GPC, scheduling and dispatching of tasks, input/output services, task synchronization and communication services, and resource management. It also is responsible for maintaining the overall integrity of the processing site in the presence of faults. This involves fault detection, isolation, and reconfiguration (FDIR). The local operating system performs all of the redundancy management functions including FDIR, background self tests, transient and hard fault analysis, and fault logging.

The services provided by local operating systems at various processing sites are similar although they may differ in implementation. For example, the multiprocessor version of the operating system must take into account the multiplicity of processors for task scheduling. Similarly, it must also consider the more complex task of redundancy management and cycling of spare units. The uniprocessor operating system can also have different variations depending upon the level of redundancy and the I/O configuration.

The Local Operating System in each computer interfaces with the Network Operating System. The Network Operating System is responsible for system level functions. These include an orderly start and initialization of various buses and networks, communication between processes executing in different computers, system level resource management, and system level redundancy management. System level resources are the GPCs, the I/O, IC, and the MM buses, and the shared data and programs stored in the mass memory or in some other commonly accessible location. System redundancy management includes FDIR in the I/O and IC node networks, correlation of faults in GPCs (both transient and hard faults), reassignment of computers to functions (function migration), and graceful degradation in case of a loss of a processing site.

Some of the functions of the Network Operating System are centralized the Global Computer. The Global Computer is responsible for system start, resource management, redundancy management, and function migration. It needs status knowledge of all processing sites and it must be able to command other GPCs to perform specific functions. This communication is accomplished via the Network Operating System, a portion of which is resident in each computer. The Global Computer does not participate in every system level transaction. Some of the system level functions performed by the Network Operating System may involve only a pair of nonglobal GPCs.

One of the GPCs is designated to be the Global Computer at the system bootstrap time. However, this designation can be changed during system operation by an internal or an external stimulus. The current Global Computer responds to the internal and external stimuli. It decides when control should be transferred and who the new designee should be. A potential Global candidate must receive the appropriate command from the current Global before it can assume this responsibility. There is one exception to this rule. In case of a catastrophic failure of the current Global Computer an alternate would take over as the Global. The catastrophic failure may be caused by a generic hardware failure of all the redundant channels or by a software error. To ascertain that the primary Global has failed catastrophically, all alternate global candidates listens for periodic 'heartbeats' from the Global. In the absence of such a signal for a predetermined time period the alternates go through a deterministic distributed algorithm to determine which processor assumes the role of the Global Computer.

3.1.2.5 Data Base Another important attribute of a distributed system is the treatment of the data base. The data base can be completely replicated in all subsystems or it can be partitioned among the subsystems. In addition, the data base directory can be centralized in one subsystem, duplicated in all subsystems, or partitioned among the subsystems. The AIPS approach is a combination of these.

For the mass memory data base, all GPCs will contain a directory of the MMU contents. This can be implemented as a 'directory to the directory' in order to limit the involvement of GPCs in the directory change process. The MMU directory will be static over extended intervals.

The data base that reflects the global system state will be maintain ed by the Global Computer in its local memory. A copy will be maintained by any alternate Global Computer, also in local memory.

The data base that reflects the distribution of functions among GPCs will be contained in all GPCs.

3.1.2.6 Fault Tolerance There is a considerable amount of hardware redundancy and complexity associated with each of the elements shown in Figure 1 on page 4. This redundancy allows each hardware element to be reliable, fault tolerant, and damage tolerant. From a software viewpoint, however, this underlying complexity of the system is transparent. This is true not only in the context of the applications programs but for most of the operating system as well; however, those elements of the operating system that are concerned with fault detection and recovery and other redundancy management functions have an intimate knowledge of the underlying complexity.

Hardware redundancy in the AIPS is implemented at a fairly high level, typically at the processor, memory, and bus level. There are two fundamental reasons for providing redundancy in the system: one, to detect faults through comparison of redundant results, and two, to continue system operation after component failures. Processors, memories, and buses are replicated to achieve a very high degree of reliability and fault tolerance. In some cases coded redundancy is used to detect faults and to provide backups more efficiently than would be possible with replication.

The redundant elements are always operated in tight synchronism which results in exact replication of computations and data. Fault detection coverage with this approach is one hundred per cent once a fault is manifested. To uncover latent faults, temporal and diagnostic checks are employed. Given the low probability of latent faults, the checks need not be run frequently.

Fault detection and masking are implemented in hardware, relieving the software from the burden of verifying the correct operation of the hardware. Fault isolation and reconfiguration are largely performed in software with some help from the hardware. This approach has flexibility in reassigning resources after failures are encountered, and yet it is not burdensome since isolation and reconfiguration procedures are rarely invoked.

3.1.2.7 Damage Tolerance One of the AIPS survivability related requirements is that the information processing system be able to tolerate those damage events that do not otherwise impair the inherent capability of the vehicle to fly, be it an aircraft or a spacecraft.

The requirement for damage tolerance will be applied to redundant GPCs, intercomputer communications, and to communication links between GPCs and sensors, effectors, and other vehicle subsystems.

The internal architecture of the redundant computers supports the damage tolerance requirement in several ways. The links between redundant channels of a computer are point-to-point. That is, each channel has a dedicated link to every other channel. Second, these links can be several meters long. This makes it possible to physically disperse redundant channels in the target vehicle. The channel interface hardware is such that long links do not pose a problem in synchronizing widely dispersed processors.

For communication between GPCs and between a GPC and I/O devices a damage and fault tolerant network is employed. The basic concept of the network is as follows.

The network consists of a number of full duplex links that are interconnected by circuit switched nodes to form a conventional multiplex bus. In steady state, the network configuration is static and the circuit switched nodes pass information through them without the delays which are associated with packet switched networks. The protocols and operation of the network are identical to a multiplex bus. Every transmission by any subscriber on a node is heard by all the subscribers on all the nodes just as if they were all linked together by a linear bus.

The network performs exactly as a virtual bus. However, the network concept has many advantages over a bus. First of all, a single fault can disable only a small fraction of the virtual bus, typically a link connecting two nodes, or a node. The network is able to tolerate such faults due to a richness of interconnections between nodes. By reconfiguring the network around the faulty element, a new virtual bus is constructed. Except for such reconfigurations, the structure of the virtual bus remains static.

The nodes are sufficiently smart to recognize reconfiguration commands from the network manager which is one of the GPCs. The network manager can change the bus topology by sending appropriate reconfiguration commands to the affected nodes.

Second, weapons effect induced damage or other damage caused by electrical shorts, overheating, or localized fire would affect only subscribers in the damaged portion of the vehicle. The rest of the network, and the subscribers on it, can continue to operate normally. If the sensors and effectors are themselves physically dispersed for damage tolerance or other reasons and the damage event does not affect the inherent capability of the vehicle to continue to fly, then the control system would continue to function in a normal manner or in some degraded mode as determined by sensor/effector availability. The communication mechanism, that is, the network itself, would not be a reliability bottleneck.

Third, fault isolation is much easier in the network than in multiplex buses. For example, a remote terminal transmitting out of turn, a rather common failure mode, can be easily isolated in the network through a systematic search where one terminal is disabled at a time. This, in fact, is a standard algorithm for isolating faults in the network.

Fourth, the network can be expanded very easily by adding more nodes. In fact, nodes and subscribers to the new nodes (I/O devices or GPCs) can be added without shutting down the existing network. In bus systems, power to buses must be turned off before new subscribers or remote terminals can be added.

Finally, there are no topological constraints which might be encountered with linear or ring buses.

3.1.2.8 Source Congruency An important consideration in designing AIPS is the interface between redundant and simplex elements. This interface design is crucial in avoiding single point faults in a redundant system. One must perform source congruency operations on all simplex data coming into a redundant computer. It is not sufficient to distribute simplex data to redundant elements in one step. The redundant elements must exchange their copy of the data with each other to make sure that every element has a congruent value of the simplex data. The AIPS architecture not only takes this requirement into account but also provides efficient ways of performing simplex source congruency through a mix of hardware and software. The simplex to redundant interface is also the place where the applications programmer gets involved in the processor redundancy and the applications code complexity multiplies. The AIPS processor level architecture is designed such that it separates the source congruency and computational tasks into two distinct functional areas. This reduces the applications code complexity and aids validation.

A description of simplex source congruency is contained in Section 6.3.

3.1.2.9 Mass Memory The mass memory in AIPS provides the following capabilities.

(1) System Cold Start/Restart.

(2) Function Migration Support.

(3) Overlays for local memory of General Purpose Computers.

(4) System Table Backup.

(5) Storage for system-wide common files.

(6) Program Checkpointing.

These requirements are discussed in the following.

To support the first requirement, the mass memory should be able to hold a copy of all the programs and data necessary to start the system. These include all those programs that are not in the nonvolatile (PROM) memory of the General Purpose Computers. For cold start, each GPC loads its volatile program memory (RAM) from the mass memory. An initial version of the system tables for bootstrapping the system will also be in the mass memory.

The second requirement implies that a copy of migratable functions (code) is present in the mass memory. Typically, the destination GPC will read this copy into its local memory unless it happens to already have the code resident. Mass memory may also be used to transfer data associated with the migrating function. The source GPC would write the data into the mass memory and the destination GPC would retrieve the data from the mass memory.

The memory overlay requirement implies that some code that is needed in later phases of the mission will be initially resident only in the mass memory. It will be read into local GPC memories as needed during the mission.

A copy of the system tables can be maintained in the mass memory for backup in case the primary copy is lost due to power transients.

The mass memory will also be used for storage of system-wide common files.

Finally, the mass memory may also be used to store program checkpoint data to support function restart in the same computer or function migration to another computer.

To support the functional requirements stated here it is necessary that the mass memory have sufficient speed to support large code transfers necessary for function migration. It should be nonvolatile and it should have high reliability (at least capable of masking all single faults).

There will be two parts of the mass memory: read-only and read/write. Most of the code will be stored in the read only part of the memory. Some of the code may be in read/write part to support inflight alteration of code for some applications. The data areas will mostly be read/write although some data such as display overlays will be in read-only part.

3.1.3 System Services The AIPS system software provides numerous services in support of user applications. These services include functions invoked explicitly by applications programs, as well as functions performed autonomously by the system to maintain proper operation. These services can be organized in six broad categories as shown in Figure 2 on page 11.

To facilitate growth and change, testability, system integration, and software fault tolerance, the AIPS system software will be organized as a series of layers, where each layer provides services to the layers above through well-defined interfaces. Since each layer hides implementation details from the other layers, the impact of a change which does not alter the interface is confined to the affected layer. Each layer is allowed to use only services provided by lower layers in the hierarchy. This means that if an interface is changed, it affects only the layers above. Adherence to this rule also aids integration, testing, and exception handling. Exceptions will be handled in the layer in which they occur unless this becomes impossible, in which case the problem is passed to higher layers for resolution.

Each general purpose computer (GPC) in the AIPS system has essentially the same layered operating system, except that one of the computers, designated the Global Computer, performs certain additional global functions. Most of the upper layers of the operating system have global functions associated with them, as will be described. The global functions are not time critical for the operation of the AIPS system, and can be migrated to another GPC if necessary.

The implementation of the lowest layer of the operating system for the multiprocessor will be substantially more complicated than that for the uniprocessor, although the services provided to the user will be similar. The operating systems may also vary with respect to the detailed I/O configurations that they support.

The AIPS system services are now described briefly, starting at the bottom of the hierarchy. Section 3.5.2 of this document describes the system software in greater detail.

3.1.3.1 Local Computer Management This layer comprises essentially the local operating system of each general purpose computer (uniprocessor and multiprocessor). It provides a reliable underlying machine for each of the layers above, and hides the details of local computer resource management.

At this layer, the AIPS system provides the following user services:

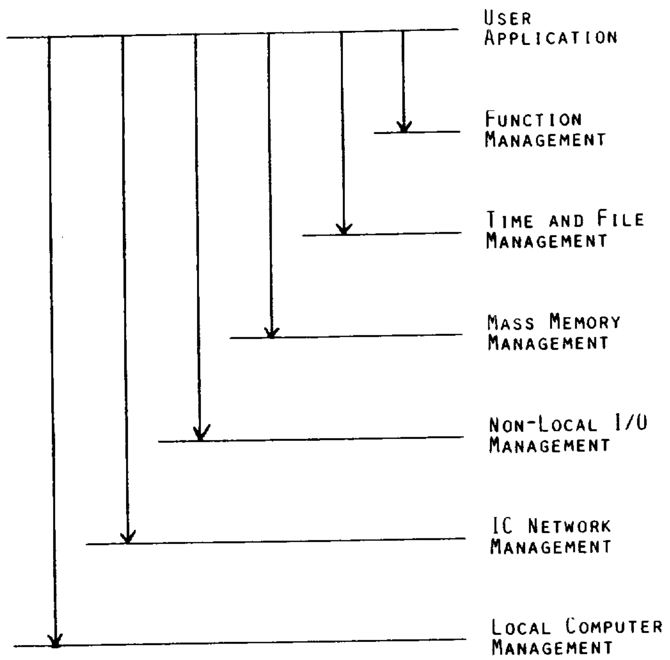

Figure 2. System Services

- <u>Function execution</u> - task scheduling, dispatching, suspension, and termination. Scheduling is based on priority, time occurrence, or event occurrence.
- <u>Local memory management</u> - dynamic allocation and deallocation for each task.
- <u>Local intertask communication</u> - supporting parameter passing as well as mailboxes.
- <u>Exception handling</u> - both user-defined and system-defined exceptions are recognized. If requested, AIPS will pass control to a user-designated routine when an exception occurs.
- <u>Local I/O</u> - manages local I/O buses and dedicated I/O.

In addition, the AIPS system will initialize and maintain reliable operation of the local GPC. This includes self test; hardware failure detection, identification, and reconfiguration (FDIR); and processor initialization, synchronization, and restart.

3.1.3.2 IC Network Management This layer and the layers above comprise (essentially) the network operating system portion of the AIPS system software. This layer provides a network of GPCs, capable of reliably communicating with each other via the intercomputer bus, while hiding the details of that communication.

At this layer, AIPS provides the following user service:

- <u>Interfunction communication</u> - enabling user functions to communicate with each other without needing to know which GPC the functions are located on. It supports parameter passing as well as mailboxes.

In addition, the system will initialize and maintain reliable operation of the IC network. This includes self test, a periodic subscriber poll, network FDIR, and network initialization. These functions are global, except that each GPC participates in failure detection, and responds to global requests.

3.1.3.3 Nonlocal I/O Management This layer is responsible for the functioning of the global and regional I/O networks, configuring and managing them as I/O buses, while hiding the details of I/O management.

At this layer, AIPS provides the following user service:

- Input/output - access to the devices of the global and regional I/O buses. Certain generic device drivers will be available, while the user may have to develop drivers for specialized devices.

In addition, AIPS will initialize and maintain reliable operation of the global and regional I/O networks. This includes self test, periodic status polls, network and device FDIR, and network initialization. The global computer manages the global I/O network, and designates a manager for each regional I/O network. Each GPC participates in failure detection, and responds to requests from the global and regional managers.

3.1.3.4 Mass Memory Management This layer provides an AIPS system with a mass memory capability for storing programs and data. It provides the following service:

- Access to the mass memory - allows each GPC to use the mass memory to support AIPS system functions and user applications.

In addition, AIPS must initialize and maintain reliable operation of the mass memory, including self test, FDIR, and contention resolution.

3.1.3.5 Time and File Management This layer provides a reliable network of communicating GPCs having a common time base and file system. The details of time distribution and file system implementation are hidden. It provides the following user services:

- System time - user functions on each GPC will be able to determine system time with granularity and skew as specified in the section on performance goals.
- Basic file operations - create, delete, open, close, read, write; the file is used as an abstraction to indicate the transfer of sequences of data values between a user program and external sources or destinations which may include the mass memory, local memory, and input/output devices. The file services to be provided are those that support the use of files in the Ada[1] language.

1 Ada is a registered trademark of the U.S. Department of Defense (AJPO).

The time function is distributed in that each GPC must maintain its local time aligned to the system time. The global computer selects the source for system time, broadcasts its value, and performs the associated FDIR. In addition, AIPS initializes the file system, maintains file directories, and provides for file system FDIR.

3.1.3.6 Function Management This layer manages the assignment of functions to processors in the AIPS system, while hiding the details of the assignment process. It provides the following user services:

- Initial function assignment - supporting initial program loading.
- Pre-planned reassignment - based on mission phases, events, or time.
- Pre-planned function migration - in response to failure or repair events.

The user must ensure that the preplanned assignments, reassignments, and migrations are feasible and are supported by the system I/O configuration. The AIPS system will raise an exception if an infeasible migration is requested.

These functions are under global control, although each GPC is responsible for carrying out global requests.

3.1.4 Proof-of-Concept System To demonstrate feasibility of the Advanced Information Processing System concept described in the preceding sections, a laboratory proof-of-concept (POC) system will be built. The major ideas and concepts that are novel to the AIPS and that need to be demonstrated in the laboratory system are as follows.

(1) A multiprocessor that is both fault as well as damage tolerant.

(2) A uniprocessor that is fault and damage tolerant.

(3) An intercomputer communication medium (I/O Network) that is fault tolerant, damage tolerant, and has demonstrable growth capability.

(4) An Input/Output mechanism (I/O Network) that is reliable, and damage tolerant.

(5) A fault tolerant mass memory.

(6) Processing sites of varying redundancy levels, preferably at least one each of simplex, duplex, triplex, and hybrid redundancy.

(7) A Network Operating System that supports such requirements as function migration, system reconfiguration, flexibility, and fault tolerance.

(8) Enough processing sites to make the function migration, system reconfiguration, and I/O and IC network reconfiguration meaningful and nontrivial.

(9) A Local Operating System in each processing site to perform functions in support of the overall operation and demonstration of the system.

A system configuration that can be used to demonstrate and prove these concepts is shown in Figure 3 on page 15. It consists of five processing sites which are interconnected by a triplex circuit switched network. Four of the five GPCs are uniprocessors, one simplex, one duplex, and two triplex processors. The fifth GPC is a multiprocessor that uses parallel hybrid redundancy. The redundant GPCs are to be built such that they can be physically dispersed for damage tolerance. Each of the redundant channels of a GPC could be as far as 5 meters from other channels of the same GPC.

Each of the triplex fault tolerant processors (FTPs) and the fault tolerant multiprocessor (FTMP) interfaces with three nodes of the Intercomputer (IC) node network. The duplex and the simplex processors interface with two and one nodes, respectively.

The mass memory is a highly encoded memory that interfaces with the GPCs on a triplex multiplex bus.

The Input/Output is mechanized using a 16 node circuit switched network that interfaces with each of the GPCs on 1 to 6 nodes depending on the GPC redundancy level.

Redundant system displays and controls are driven by the Global Computer and interface through the I/O network.

Each GPC has a Local Operating System and a portion of the Network Operating System. For the proof-of-concept system, initially the FTMP will be the Global Computer.

3.1.4.1 Proof-of-Concept Building Blocks Major hardware building blocks of the AIPS Proof-of-Concept configuration are as follows:

(1) <u>Fault Tolerant Processors</u> The POC system contains one simplex, one duplex, and two triplex fault tolerant processors. Each channel of the FTPs consists of a computational part and an Input/Output part. The first part contains a computational processor (CP), memory, timers and clocks. The second part contains an Input/Output processor (IOP), memory, timers and clocks. The IOP interfaces with the I/O and the IC nodes. The CP and the IOP have a shared interface to the mass memory. The redundant processors are tightly synchronized using a fault tolerant clock. Data amongst redundant channels is exchanged on point to point links such that the system is protected against simplex source faults. The data exchange hardware also performs the fault detection and masking functions. Apart from redundancy, there are other features that provide hardware and software fault tolerance. These include watchdog timers, processor interlocks, virtual memory, and a privileged operating mode.

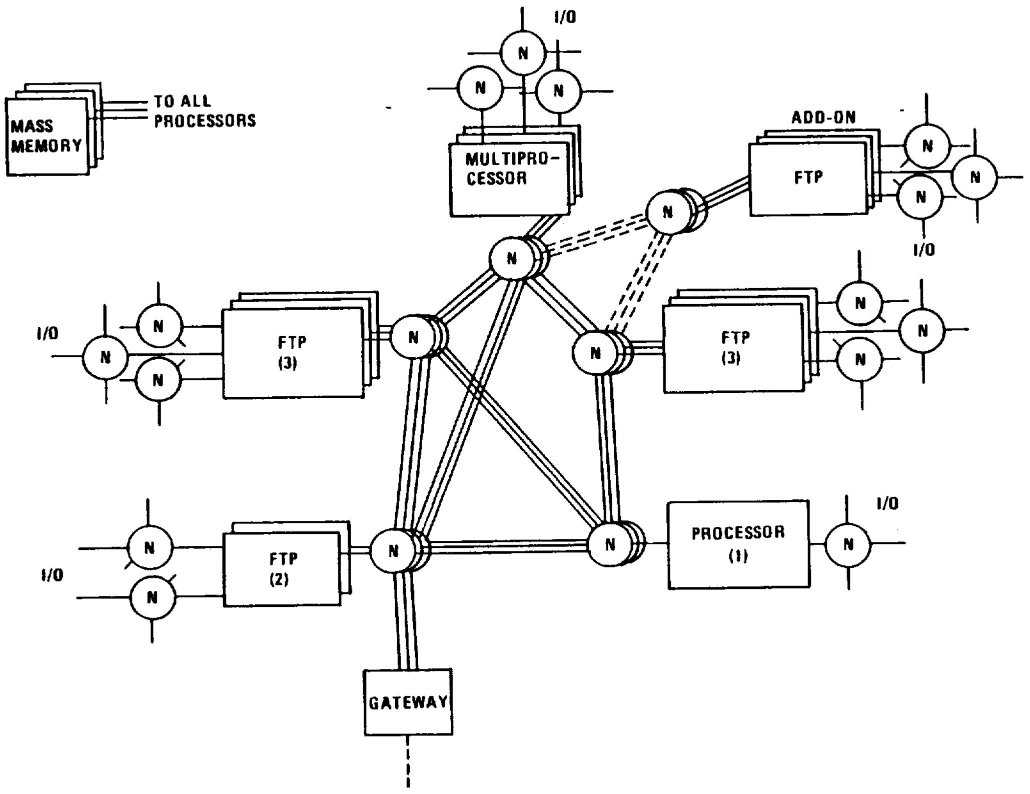

Figure 3. AIPS Proof-of-Concept Configuration

(2) Fault Tolerant Multiprocessor The POC system contains one fault tolerant multiprocessor. The FTMP has processors which are similar to the FTP processors. Each processor has its own local memory, timers, and clocks. Processors are organized in triads. Each triad consists of three processors. Members of a given triad operate in tight synchronism and fault detection and masking are performed in hardware. Dynamic reconfiguration is used to rearrange processor triads following a failure. Hardware support is provided for this dynamic reconfigurability and parallel hybrid redundancy. All processor triads have access to a shared memory. Shared memory organization is similar to processor organization. The processor-shared memory communication is on a fully cross-strapped bus. This reduces bus contention and provides damage tolerance. The FTMP interfaces with the external world through the IC Network nodes, the I/O Network nodes, and the Mass Memory buses. Not all processors in the FTMP have all of these interfaces. The details of the proof-of-concept configuration of the FTMP, such as number of processors, shared memory modules, and their external interfaces, will be determined during the preliminary design. The

fault tolerance features in the FTMP are similar to those described for the FTP.

(3) Intercomputer Network The intercomputer network consists of three identical layers of a 5-node circuit switched network. This triplex network provides for fault and damage tolerant communication amongst all GPCs. Each of the nodes in the network is a 5-ported circuit switching device. One of the ports interfaces with a subscriber GPC and the other four ports connect the node to other nodes in the same layer of the network. The three layers of the network are not connected to each other. A node can connect any input port to one or more of its output ports when commanded to do so by the Global Computer. Once all the nodes in the network have been configured by the Global Computer the network operates exactly as a virtual multiplex bus. Nodes detect protocol and transmission errors and record these for later retrieval and analysis by the Global Computer.

(4) Input/Output Network The I/O network consists of a 16-node network that is similar to one layer of the IC network. Six of these nodes interface with the FTMP, 3 each interface with the triplex FTPs, two with the duplex FTP and one with the simplex processor.

(5) Mass Memory The AIPS mass memory is a fault tolerant memory that interfaces with the GPCs on a triplex multiplex bus. The mass memory has a triplex interface unit, one for each of the three mass memory buses. Redundant address, data, and command words from the GPCs are received, deskewed, and voted by each of the three interface units. For memory read operations, each interface unit responds with data on one mass memory bus. The requesting GPC votes on the redundant data received on the triplex bus. Most of the mass memory is read-only while some of it is read-write.

3.1.4.2 Architecture of AIPS Building Blocks The following sections describe the architecture of each of the AIPS building blocks.

3.1.4.2.1 Fault Tolerant Processor The architectural description of the FTP is divided into three sections: Software View, Hardware View, and External Interfaces.

3.1.4.2.1.1 Fault Tolerant Processor: Software View The FTP or the uniprocessor architecture from a software viewpoint appears as shown in Figure 4 on page 18. The uniprocessor can be thought of as consisting of two separate and rather independent sections: the computational core and the Input/Output channel.

The computational core has a conventional processor architecture. It has a CPU, memory (RAM and ROM), a Real Time Clock, and interval timer(s). The Real Time Clock counts up and can be read as a memory location (a pair of words) on the CP bus. Interval timers are used to time intervals for scheduling tasks and keeping time-out limits on applications tasks (task watchdog timers). An interval timer can be loaded with a given value which it immediately starts counting down and when the counter has been decremented to zero, the CPU is interrupted with a timer interrupt. A watchdog timer is provided to increase fault coverage and to fail-safe in case of hardware or software malfunctions. The watchdog timer resets the processor and disables all its outputs if the timer is not reset periodically. The watchdog timer is mechanized independently of the basic processor timing circuitry.

There also appears on the processor bus a set of registers, called the data exchange registers. These are used in the redundant fault tolerant processor to exchange data amongst redundant processors. From a software viewpoint, this is the only form in which hardware redundancy is manifested.

On a routine basis the only data that needs to be exchanged consists of error latches and cross channel comparisons of results for fault detection. These operations can be easily confined to the program responsible for Fault Detection, Isolation, and Reconfiguration. Voting of the results of the redundant computational processors is performed by the Input/Output processors. Therefore, the remaining pieces of the Operating System software and the applications programs need not be aware of the existence of the data exchange registers. The task scheduler and dispatcher, for example, can view the computational core as a single reliable processor.

The other half of the processor is the Input/Output channel. The I/O channel has a CPU (same instruction set architecture as the CP), memory (RAM and ROM), a Real Time Clock, and an Interval Timer(s). This part of the I/O channel is identical to the CP except that it has less memory than the CP.

The IOP has interfaces to the intercomputer bus, one or more I/O buses, and memory mapped I/O devices. The CP and the IOP also have a shared interface to the system mass memory. These external interfaces of the FTP will be discussed in the next two sections.

The IOP and CP exchange data through a shared memory. The IOP and CP have independent operating systems that cooperate to assure that the sensor values and other data from Input devices is made available to the control laws and other applications programs running in the CP in a timely and orderly fashion. Similarly, the two processors cooperate on the outgoing information so that the actuators and other output devices receive commands at appropriate times. This is necessary to minimize the transport lag for closed loop control functions such as flight control and structural control.

The CP and IOP actions are therefore synchronized to some extent. To help achieve this synchronization in software, a hardware feature has been provided. This feature enables one processor to interrupt the other processor. By writing to a reserved address in shared memory the CP can interrupt the IOP and by writing to another reserved location the IOP can interrupt the CP. Different meanings can be assigned to this interrupt by leaving an appropriate message, consisting of commands and/or data, in some other predefined part of the shared memory just before the cross-processor interrupt is asserted.

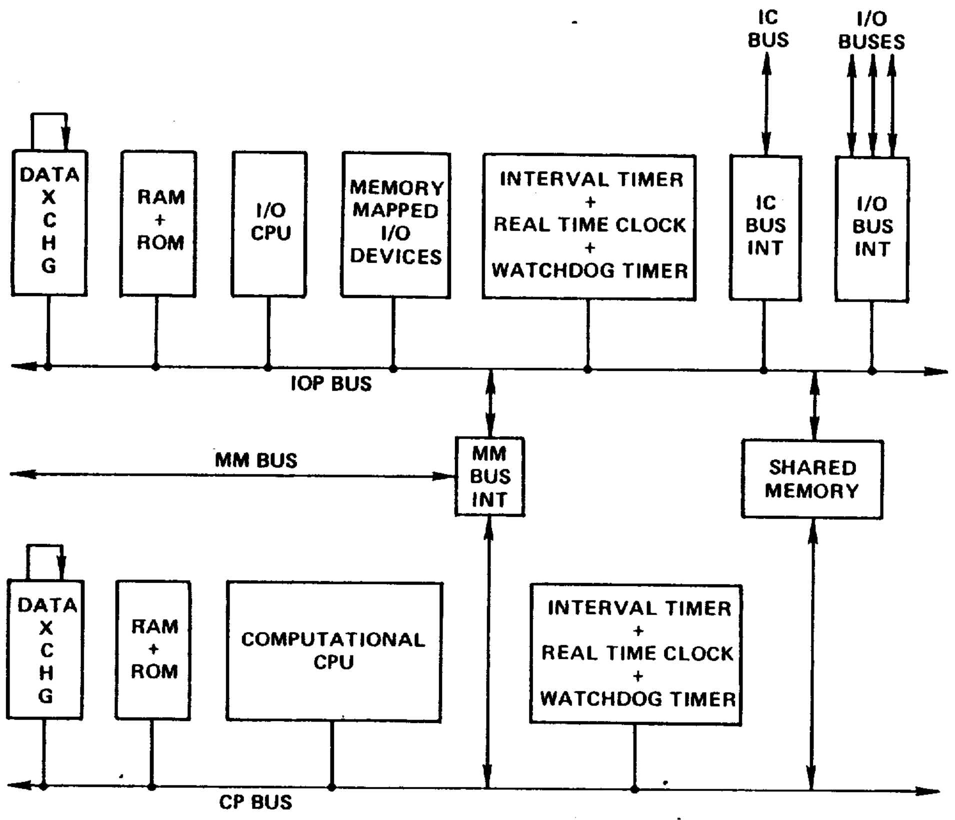

Figure 4. Fault Tolerant Processor Architecture: Software View

For routine flow of information in both directions, the shared memory will be used without interrupts but with suitable locking semaphores to pass a consistent set of data. The interrupts can be used to synchronize this activity as well as to pass time critical data that must meet tight response time requirements. In order to assure data consistency it is necessary that while one side is updating a block of data the other side does not access that block of data. This can either be implemented through semaphores in software or through double buffering. Hardware support for semaphores, in the form of test & set instruction, is provided in the IOPs and CPs.

There are many attractive features of this architecture from an operational viewpoint. The most important of these is the decoupling of computational stream and the input/output stream of transactions. The computational processor is totally unburdened from having to do any I/O transaction. To the CP all I/O appears memory mapped. And this not only includes I/O devices but also all other computers in the system as well. That is, each sensor, actuator, switch, computer, etc. to which the FTP interfaces can simply be addressed by writing to a word or words in the shared memory.

Data from other processing sites is received by each IOP on the redundant IC buses, hardware voted, and then deposited in their respective shared memories. Simplex source data such as that from I/O devices, local processors, etc. is received by the single I/O processor that is connected to the target device. This data is then sent to the other two I/O processors using the IOP data exchange hardware. The congruent data is then deposited in all three shared memory modules. In either case, the computational processors obtain all data from outside that has already been processed for faults and source congruency requirements by the I/O processors.

The data exchange mechanism appears to the software as a set of registers on the processor bus. Data exchange between redundant processors takes place one word at a time. Two types of data exchanges are possible: a simplex exchange or a voted exchange. The purpose of a simplex exchange is to distribute congruent copies of data that is available only in one channel of the FTP to all other channels. The purpose of a voted exchange is to compare and vote computational results produced by redundant processors. In the FTP architecture, these exchanges are mechanized as follows.

To perform a voted exchange, each processor writes the value to be voted in a transmit register called X_V. Writing to this register initiates a sequence of events in hardware which culminates with the voted value being deposited in the receive register of each processor. The processor can read the receive register at this point to fetch the voted value. The whole transaction takes of the order of 5 microseconds. The hardware is designed to lock out access to the receive register while the exchange is in progress. If the processor tries to read the receive register before the transaction has completed, the processor hangs up. As soon as the data becomes available, the processor is released and the register read cycle completes normally. The processor wait is transparent to the software. It is not necessary to time the interval between writing of the transmit register and reading of the receive register in software. The two operations can be performed as a sequence of two instructions without an intermediate wait.

To perform a simplex exchange, the data to be transmitted is written to one of the simplex transmit registers. In the triple redundant version of the FTP there are three such registers. They are called X_A, X_B, and X_C. X_A is used to transmit simplex data from channel A to all others. Similarly X_B transmits data from B and X_C transmits data from C. Writing to one of these registers initiates a sequence of events in hardware which culminates with a congruent copy of the data word being deposited in the receive register of each processor. The receive register can be read at this point by each processor to fetch the congruent copy of the simplex data.

It has been pointed out earlier that the software appearance of the redundant FTP is the same as that of a simplex processor. All redundant processors have identical software and execute identical instructions at all times. This architecture is carried forth in the data exchange hardware and software as well. The data exchange hardware is designed such that all redundant processors execute identical instructions when exchanging data. As an example, consider a simplex source transmission from channel A. Assume that channel A has a sensor value in its internal memory location, called MEMORY, that it needs to send to channels B and C. This requires execution of the following sequence of four instructions:

```
1     LOAD  RO,MEMORY
2     STORE RO,X_A
3     LOAD  RO,X_R
4     STORE RO,MEMORY
```

The data to be transmitted is fetched from memory (instruction 1) and written to transmit register X_A (instruction 2). All three processors execute these instructions. However, only processor A's value is transmitted to the receive register of A, B, and C. Transmissions from B and C are ignored by the hardware. This will be explained in the next section which deals with the FTP architecture from a hardware viewpoint. In instruction 3 all processors read their receive register (X_R) to accept the congruent value of the data transmitted by A. In instruction 4 this value is transferred to an internal memory location.

Voted data exchange requires a similar sequence of instructions. The only difference is that in instruction 2, rather than storing the value in one of the simplex transmit registers, it is stored in the voted exchange register, X_V.

3.1.4.2.1.2 Fault Tolerant Processor: Hardware View The triplex FTP architecture from a hardware viewpoint appears as shown in Figure 5 on page 22. There are three identical hardware channels. Each channel has a computational processor, an I/O processor, and some hardware that is shared by the CP and the IOP. The internal details of the CP and the IOP such as the CPU, memory, timers, etc., have been described in the preceding section. They are not shown in Figure 5 so that other details such as the redundancy dimension be shown more clearly.

The common hardware consists of a shared memory, the data exchange registers, and the mass memory interface. The shared memory is used exchange information between the CP and the IOP while the data exchange registers are used to exchange information between redundant copies of the CP or IOP. Common hardware access conflicts between the CP and the IOP are resolved by a bus arbitrator. The bus arbitration logic is designed such that each channel resolves the conflict in favor of the same processor (that is, either the IOP or the CP) deterministically. This is necessary to maintain tight synchronism between redundant copies of processors. This is only one of several conditions necessary for synchronous operation. Stated in more general terms, two hardware conditions are necessary. First of all there should be a common time base that is used by all channels for timing events. Second, all timing events should be deterministic in nature. If these two hardware conditions are met, the redundant channels can be synchronized. Once they are synchronized, they will stay synchronized if all channels execute identical software.

To obtain a common time base, the oscillators in redundant channels are phase locked to each other. The details of this mechanization are covered in section 3.5.1.1.3.4 (FTP Fault Tolerant Clock). To assure that all timing events are deterministic, it is necessary to use a synchronous bus internal to each processor. As an example, when a CPU references a memory location on the bus the memory cycle should complete in a fixed time interval. This does not necessarily imply that all memory cycles should take the same time. It is possible in the FTP architecture to mix different types of memories, such as PROM and RAM, which may have different access times. The only necessary condition is that access to a given location always take the same length of time. This also applies to any I/O activity performed by the processor. The hardware is built such that when a processor accesses a device available in only one channel the other processors wait the same length of time.

A very important aspect of the FTP architecture is the interconnection hardware between redundant channels. This hardware serves three purposes. First of all, it provides a path for distributing simplex data available in only one channel to all other channels. Second, it provides a mechanism for comparing results of the redundant channels. And third, it provides a path for distributing and comparing timing and control signals such as the fault tolerant clock and external interrupts.

To distribute simplex data from one channel to all others without introducing single point faults in the design, it is necessary to adhere to source congruency requirements. These are explained in Section 6.3. One of these dictates that in order to tolerate single faults it is necessary to provide four fault containment regions. In the triplex FTP architecture six fault containment regions are provided. The triplex processor provides the basic three fault containment regions. Three additional regions are provided in the form of interstages which receive data from processors and rebroadcast them back to processors. The interstages are mechanized such that they have independent voltage and timing reference. This assures that faults in processors would not propogate to interstages and vice versa. Since an interstage is essentially a buffer with receivers and transmitters, it is relatively a small and simple piece of electronics. It is, therefore, much more convenient to provide three additional fault containment regions rather than just one as required for source congruency. It also makes the FTP architecture symmetric.

As explained in the preceding section, the data exchange hardware appears as a set of five registers on the processor bus. Four of these (X_A, X_B, X_C, and X_V) are the transmit registers and the fifth one is the receive register, X_R. For simplex source exchanges, say, a 'from A' exchange, data in X_A register in channel A is transmitted to the three interstages. The interstages rebroadcast this data to every processor. The three copies received by each processor are voted in hardware on a bit-by-bit basis. The voted result is deposited in X_R. For voted exchanges, each channel writes the data to be voted in X_V register. Writing to X_V results in the data being transmitted to the channel's own interstage. The second half of the operation is the same as for simplex exchange. In both cases, the exchange hardware masks any single faults while voting on three copies and also records the source of fault in an error latch. The error latch can be read by software as a memory location.

The data exchange hardware is explained in greater detail in section 3.5.1.1.3.1.

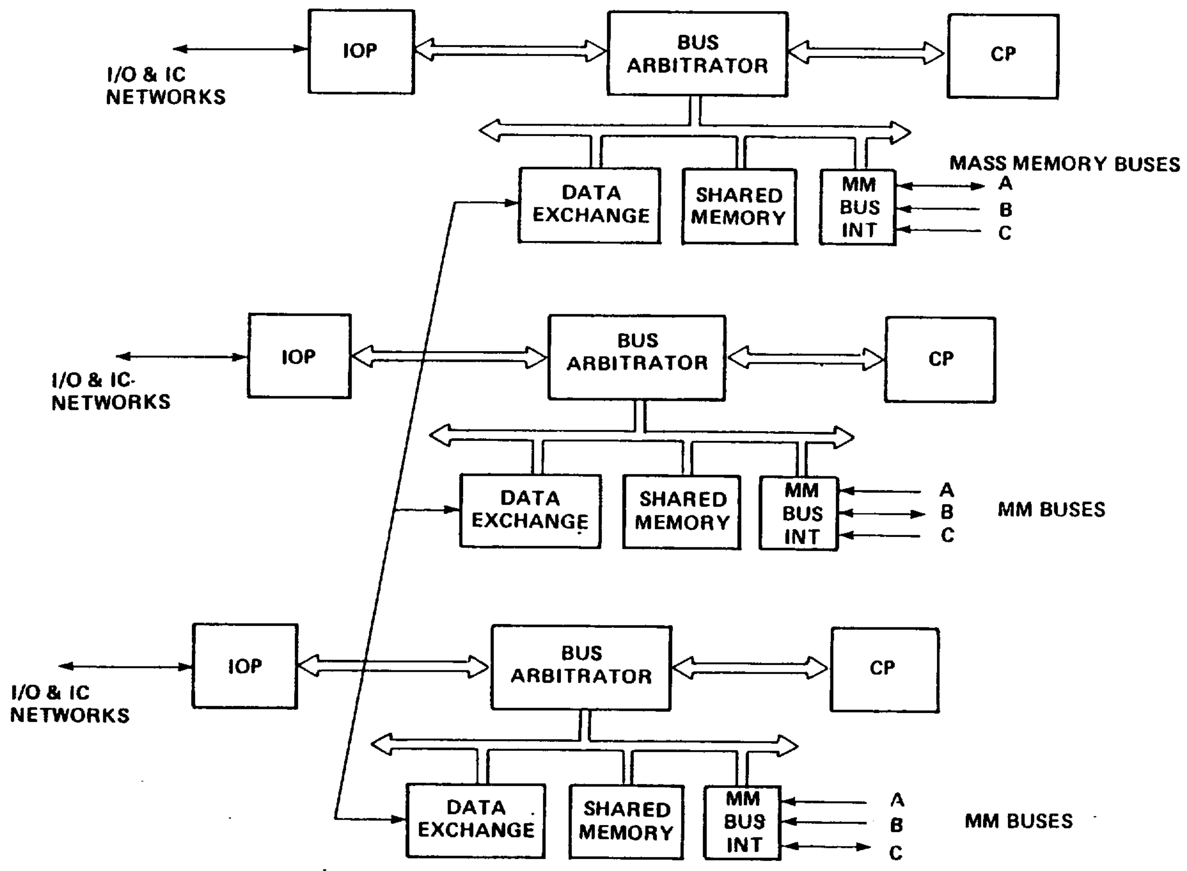

Figure 5. Fault Tolerant Processor Architecture: Hardware View

3.1.4.2.1.3 Fault Tolerant Processor: External Interfaces The external devices that interface with the FTP are the mass memory, the Intercomputer network, and the I/O network. Figure 5 on page 22 shows the interface between a triplex FTP and the triply redundant mass memory bus. This interface hardware is shared in each channel by the CP and the IOP. Each channel of the FTP is enabled on one of the three buses. The FTP transmits commands and data synchronously on three buses to the mass memory where they are received and voted in hardware. The interface hardware performs the necessary parallel to serial data conversion, appends cyclic redundancy check byte (CRC), and transmits serial data on the bus. Each processor channel listens to all three mass memory buses. Data received from the mass memory is voted in hardware on a bit-by-bit basis. Any disagreements on the mass memory bus are recorded in error latches for later analyis by software. Any CRC failures are also recorded separately. Voted data is then converted from serial to parallel format.

Figure 6 on page 24 shows the interface between a triplex FTP and the triply redundant Intercomputer Network.

The IC network interface is dedicated to the I/O processor. Other than that, it is very much like the mass memory interface. Each IOP listens to all three IC networks but can transmit on only one IC network. The IC network interface circuitry is responsible for parallel to serial data conversion, resolving network contention, and transmitting data on the network. In the other direction, the network interface hardware listens to three bit streams, deskews them, votes and masks single faults and converts the voted bit stream from serial to parallel. It stores any disagreements on the networks in error latches or registers. Non compliance with the network protocol is also recorded separately for each of the three networks.

An IOP also interfaces to one or more I/O networks. This interface is different from the IC network interface to the extent that the I/O networks may not be redundant. Redundant I/O networks interface with the FTP the same way as the IC network nodes. For simplex I/O networks, it is necessary that when the processor is communicating with an I/O device all other processors execute the same software and wait identical lengths of time to stay synchronized. Also, any data received from the I/O devices must be distributed to all other processors using the data exchange registers. Although an I/O network may not be redundant, an FTP may have more than one connection to an I/O network through multiple IOPs.

An I/O network may be dedicated so that only those I/O devices that are used solely by this FTP are on this network. Or the I/O network may be a shared network that connects multiple computers to shared I/O devices.

Finally, an IOP may also have local dedicated I/O devices that can be accessed directly by the IOP as memory locations. The memory mapped I/O may consist of local switches, discretes, A/Ds, D/As and interrupt driven devices. This interface differs from dedicated I/O bus interface in the sense that I/O signals on the bus may be already conditioned and processed by a local processor and the IOP interfaces through this bus to the local processor which may control a number of I/O devices.

Although it is possible to interface interrupt driven I/O devices to the IOPs, none will be included in the proof-of-concept system.

An IOP may transmit on an IC bus, a shared I/O bus, or a dedicated I/O bus only if it is enabled to do so by a majority of the IOPs. An IOP can also disable itself any time.

The interface of an FTP to the IC buses is somewhat different if the FTP redundancy level is not the same as the IC redundancy level. For example, if the FTP is a duplex system rather than a triplex then there will be only two IOPs, one IOP per channel. Each IOP will listen to and vote on all three IC buses and it will transmit on one IC bus. Transmissions from duplex processors will, therefore, be heard only on two out of three buses. Similarly, simplex processors will listen to all buses but will transmit on only one bus. Voting logic in the bus interface circuits is suppressed when the transmitting processor is simplex. It is necessary to rely on compliance with bus protocol to detect errors.

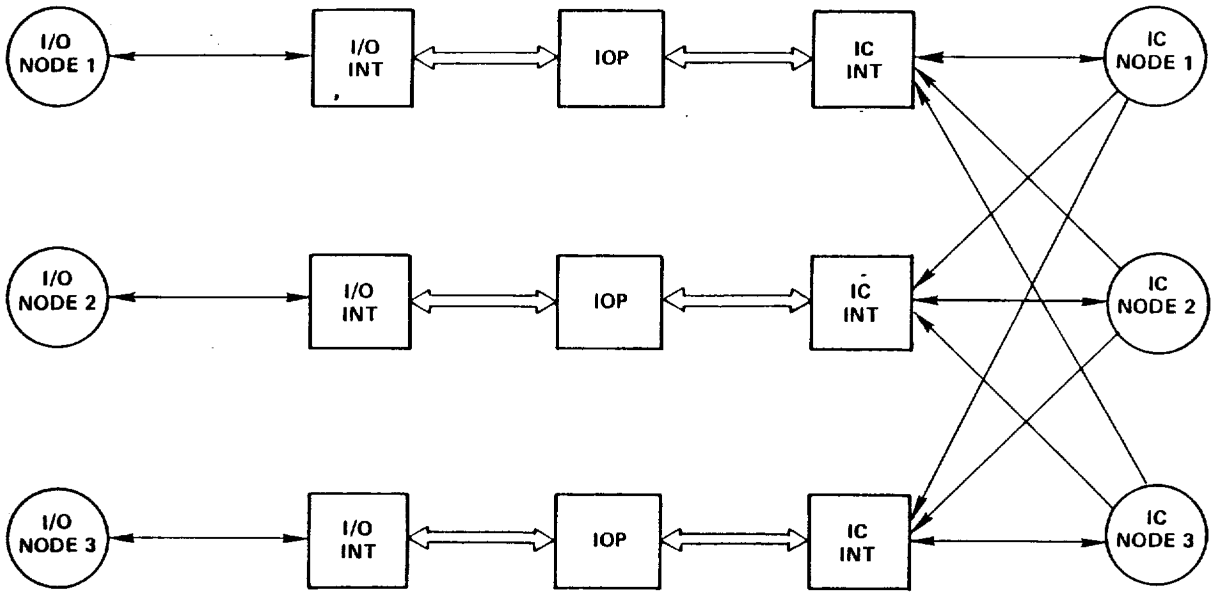

Figure 6. FTP Interface to IC and I/O Networks

The next topic of discussion is the interface of the system in degraded mode. The FTP can degrade in 3 ways, failure of an IOP, failure of a CP, and the failure of the shared hardware.

When an IOP fails, the IC bus interface would degrade from triplex to a duplex configuration. Each of the other two IOPs would listen to all three IC buses but transmissions from this FTP would be heard by other computers only on two buses. At the receiving sites fault masking would be replaced by fault detection. Since there is some inherent coded redundancy in data being transmitted on each bus, one can hope to identify the second fault with fairly high coverage. This would normally be the case when communicating to a dual-redundant computer on the network under normal circumstances.

The other effect of the failure of an I/O processor is the loss of I/O devices attached to that processor. Actually only those I/O devices that were connected only to the failed processor would be lost. If the I/O devices are cross-strapped to other IOPs through the I/O bus, for example, they would still be accessible via the other I/O processors.

Finally, the loss of an IOP also means that the CP attached to that IOP does not have access to any inputs. In other words, failure of any element in a channel can be considered the same as failure of the whole channel. That is, if an IOP, a DPM or a CP fails in a channel one could shut down all the elements in that channel of the FTP. Although this is the most convenient way to operate the system from a system software viewpoint, one can use more sophisticated strategies to obtain reliability from the system. This, however, is achieved at the expense of more complex system software.

Specifically, if an IOP fails then the corresponding CP loses access to data being provided by that IOP through the shared memory. But since the CPs are cross-strapped to each other through data exchange hardware it is possible to provide the target CP the voted value of the inputs to the other two CPs. The same can be done on the IOP side when a CP fails. If the shared memory fails either side can get around the fault by cross-exchanging data. This will be the POC operational strategy.

3.1.4.2.2 Fault Tolerant Multiprocessor The following two sections describe the software appearance of the multiprocessor and the redundancy and fault tolerance dimension of the machine, respectively.

3.1.4.2.2.1 Multiprocessor Software Appearance The multiprocessor discussion is divided into three parts: processors, shared memory, and the external interfaces. These three parts are discussed in the following sections.

3.1.4.2.2.1.1 Processors The multiprocessor architecture from a software viewpoint appears as shown in Figure 7 on page 26. This figure does not show the redundancy dimension of the computer. FTMP, from a software viewpoint, appears as a conventional, homogeneous multiprocessor. There are a number of processing elements that have access to a common memory, called the shared memory. Each processor has a local memory which is composed of Read Only Memory (ROM) and Read/Write Memory. Although not all of the programs are always resident in the local memory, they must be loaded there from the shared memory before they can be executed. The local PROM is used to hold the bootstrap loader, cold start and restart programs, frequently executed parts of the operating system, and high frequency applications programs. All other programs are loaded into the RAM on a demand basis.

Each processor has internal interval timer(s) that can be set to any 16-bit value under program control. The interval timer decrements this count and interrupts the processor when the count reaches zero. Interval timers can be used for scheduling highest frequency tasks and also as task watchdog timers.

Communication between processors can be via the shared memory or it can be via Interprocessor Communication (IPC) Buffers. The shared memory path is the slower of the two paths. The sender can write the message in the shared memory, but it does not arrive at its destination until the receiving processor reads its mail box. An alternative to this is the IPC Buffer. A processor can write to any other processor's buffer. The receiving processor is interrupted when one of the buffer locations is written into. The receiving processor's IPC Interrupt Handler can then read the message in its buffer. The buffer length is of the order of 16 words. That is, it is not a large memory array. For large data transfers, shared memory would still be used although one could pass a shared memory pointer using the IPC mechanism.

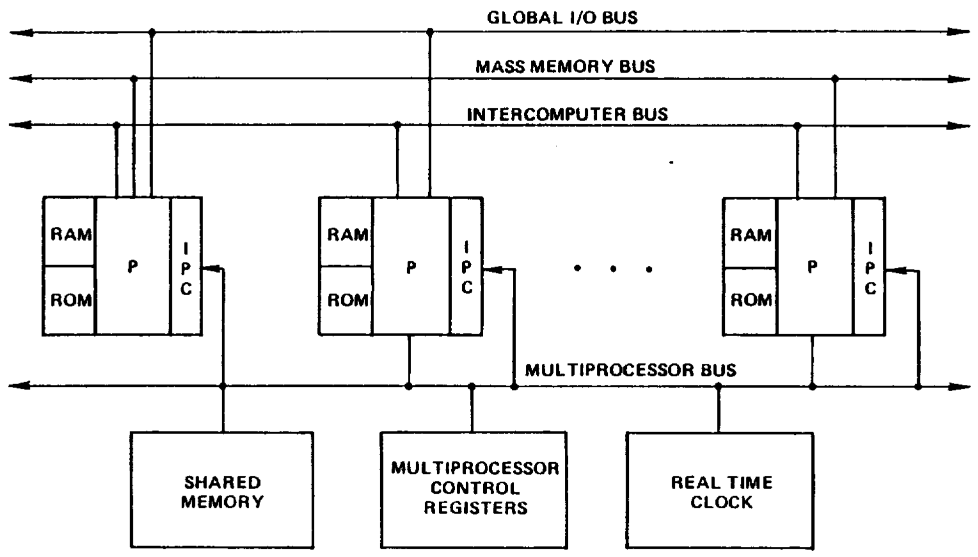

Figure 7. FTMP Architecture: A Software View

3.1.4.2.2.1.2 Shared Memory The shared memory can be accessed by the processors using the multiprocessor bus. Contention for the bus amongst multiple processors is resolved in hardware. The hardware also provides the capability to test and set a word in the shared memory in a single atomic operation. Access to data elements that are shared amongst multiple processes can be limited to one process at a time by associating a lock or semaphore with each shared data set.

Although the shared memory appears as a single monolithic unit to the software, it should be noted here that it is, in fact, composed of several segments. Such a segmented shared memory, coupled with dedicated buses from processors to memory units, makes it appear as a multiported memory. All segments of the shared memory can be accessed simultaneously if different processors happen to request access to different segments. This feature should be taken into account when locking data sets in the shared memory.

There are several other elements of the multiprocessor that can be accessed by the processors using the memory bus. One of these is the Real Time Clock (RTC). The Real Time Clock is a 32-bit counter that counts up. The RTC can be set to a given value by writing to its 'memory' address. It can be read as a two word value on the memory bus also. Reading the high order word automatically latches the low order word so that the 32-bit value of the RTC is read as a consistent set.

IPC buffers also appear as shared memory addresses as far as write operations are concerned. They can only be written to on the multiprocessor bus. They can be read only by the host processor on the internal processor bus.

Other registers in the shared memory address space can be grouped under the heading of Multiprocessor Control Registers. Their functions include:

(1) Memory Relocation (Write Only): This assigns a shared memory module to a given address space.

(2) Triad Identification (Write Only on multiprocessor bus, Read Only on processor bus): This assigns a processor to a triad.

(3) CPU Control (Write Only on multiprocessor bus, Read Only on processor bus): This controls various CPU operations such as reset, go, etc.

(4) Bus Selection (Write Only on multiprocessor bus): This tells each processor and memory which buses to listen to.

(5) Error Latch (Read Only on system bus): It records disagreements on the multiprocessor bus.

3.1.4.2.2.1.3 External Interfaces External interfaces of the FTMP consist of interfaces to other GPCs through IC nodes, interface to I/O devices through I/O nodes, and interface to Mass Memory through Mass Memory buses. Not all FTMP processors have all external interfaces. Some processors may, in fact, have none of these interfaces. These processors perform only computational function in the FTMP.

At any given time, one processor triad is assigned to communicate on the IC network. Bus interface hardware performs a bit by bit majority vote on incoming redundant data. Each processor then should have an identical copy of the incoming data. For simplex data, the voting is bypassed and it is necessary to perform source congruency. This is done by simply writing the simplex data into shared memory. Voters in the shared memory perform majority voting on three copies of the word received from the three processors. The voted data then becomes the congruent value of the simplex word.

Mass memory interface is functionally identical to the IC network interface.

The FTMP also has interfaces to one or more I/O networks. The I/O networks may or may not be redundant. One of the processor triads, with approriate I/O interfaces, is assigned to the task of managing I/O devices.

3.1.4.2.2.2 Multiprocessor Redundancy & Fault Tolerance Features As alluded to in the previous section, there is a considerable amount of complexity in the hardware that is largely transparent to the software but is responsible for making the machine fault tolerant. This complexity arises from two related features of the multiprocessor. One, every element in the system is replicated to some level. Every major element is at least triplicated, and some have even a higher level of redundancy. Two, all redundant operations must be compared to detect faults and to mask faults where appropriate. The fault detection and masking requires considerable amount of interconnections between redundant elements. These two attributes of the machine, viz., redundancy and intercommunication, are largely responsible for the multiprocessor complexity.

However, the redundant hardware elements are organized in such a fashion that this complexity is not carried over into the software. In fact, the other attribute of the machine, viz., the interconnection of the redundant elements, is what makes the hardware complexity transparent to the user. This should become clearer as the hardware architecture is described in the following sections.

3.1.4.2.2.2.1 Processors Processor in the multiprocessor, CPs or IOPs, operate in groups of three, called triads. The three members of a triad are tightly synchronized using a fault tolerant clock. (The clock operation is described in another section.) Processor organization is such that any three processors can be formed into a triad with some exceptions such as the constraints that may be introduced by packaging and external bus interface considerations. Other than these constraints, a processor element can be used as a member of a CP triad or an IOP triad. A processor may be a member of a CP triad at one time and it may be a member of an IOP triad at some other time.

Once all the available processors have been formed into CP and IOP triads, the remaining processors (spares), can be used to 'shadow' normally operating triad members. A shadow processor is tightly synchronized with the three active members of the triad and executes the same instructions as the active members. It listens to all the buses to obtain the same input data as the active members. However, it is not enabled to transmit on any bus.

Each processor has a number of control registers which are either processor specific or triad specific. All of these registers have a 'Write To' address that is an extension of the shared memory address space. They also have a 'Read From' address that is an extension of the processor's local memory address space. An example of a triad specific register is the IPC (Interprocessor Communication) Buffer. An example of a processor specific register is the Triad Identification (ID) Register. Addresses of processor specific registers have a Processor ID field in the 'Write To' address. Addresses of triad specific registers have a Triad ID field in the 'Write To' address.

There are a number of other system control and status registers that are accessed in a manner similar to the processor control registers. Examples of these will be given where relevant.

3.1.4.2.2.2.2 Shared Memory The shared memory in the multiprocessor is triplicated and operates as one contiguous triad. Physically, it is partitioned into several smaller segments and the level of replication is at the segment level. A processor and a shared memory segment are packaged together in a box or Line Replaceable Unit (LRU). They share such items as the fault tolerant clock, LRU power supply, etc.

Associated with each memory segment is a Memory Relocation Register (MRR). This register is analogous to the Processor Triad ID Register in that it allows one to identify the memory triad to which a memory module belongs. The MRR forms the high order part of the address to which the associated memory module responds. The MRR itself has an LRU specific address and can be written to on the multiprocessor bus. Thus, one can group any three memory modules to form a shared memory triad by relocating them to the same address space.

3.1.4.2.2.2.3 Processor-Memory Interface The processors and shared memory segments are fully cross-strapped. Each box or LRU containing a processor and a shared memory is connected to every other LRU using dedicated buses. The processor-shared memory communication works as follows.

Assume there are N LRUs in the multiprocessor. When a processor triad wishes to write shared memory, each member broadcasts appropriate address, data, and commands on its transmit bus. Each member of the target memory triad receives three copies of address, etc. It selects 3-out-of-N transmit buses to listen to this processor triad using information from Bus Select registers. The three copies are then voted by each member of the memory triad and appropriate action (such as storing the data) taken. Any disagreements are latched in the Processor Error Latches indicating the identity of the disagreeing processor.

For a memory read operation, the first half of the transaction is similar to the write operation. Each member of the memory triad broadcasts the data to the three requesting processors. Each processor in the requesting triad then receives three copies of the data which it votes in hardware and also latches identity of any disagreeing memory unit in its Memory Error Latch.

As indicated earlier there are a number of multiprocessor control and status registers. More examples of these have now been cited (MRR, Error Latches, Bus Select Registers). These registers are assigned addresses in an extended shared memory address space. They can be accessed by any processor triad just as if they were shared memory locations. The interface hardware to select Transmit Buses, vote on incoming processor requests, etc. can be the same hardware that is used to access the shared memory in a given LRU. This not only saves hardware but also makes use of the existing processor-memory 'bus' that cross-straps all LRUs. In fact, this same communication medium can be used to write to the Interprocessor Communication Buffers. Such an arrangement also makes the software appearance of the machine rather straightforward.

3.1.4.2.2.2.4 FTMP Clocking The tight synchronization between processor elements is maintained using standard fault tolerant clocking techniques (analog phase locked loop or digital compensation) developed previously. The multiprocessor fault tolerant clock functions as follows.

Each processor and shared memory in every LRU has a common oscillator. Four of these oscillators, called the active elements, are chosen to form the quad-redundant fault tolerant clock. (Four clock elements are necessary to tolerate all single point clock failures). Any four operating oscillators can be chosen as active elements. The active clocks are distributed to every processor and memory LRU in the system. Each active element listens to the other three active clocks and locks itself to the majority. The nonactive clocks phase lock to any three out of four active clocks. If an active elements fails, it is replaced by a previously inactive element. In other words, every clock is sent to every LRU in the system. Since the basic philosophy is to have dedicated paths rather than multiplex buses, there is a dedicated clock bus that goes from every LRU in the system to every other LRU.

Each LRU in effect listens to three active elements and synchronizes itself to the majority. This clock is then used for all internal timing events such as processor clocking, memory clocking, decrementing of the interval timer, and incrementing of the real time clock.

The real time clock is a 32-bit counter. Such a counter exists in every LRU and is accessible on the system memory bus. Real time clocks in all the LRUs respond to the same system memory address. RTC counters always respond to write (or set) requests. Thus all the real time clocks in the FTMP can be set to a given time simultaneously. Once set, they all count up at the same rate since they are clocked by the fault tolerant clock. During normal operation, three of the counters are selected to be active. There is an RTC Select Register in every LRU that determines which three RTCs to listen to. When one of the counters fails, it is replaced by another operating RTC counter by updating all RTC Select Registers in all LRUs.

3.1.4.2.2.2.5 External Interfaces The multiprocessor interfaces to the external world through three different types of buses. Interface to other General Purpose Computers is via the IC network. Interface to the I/O devices is via the I/O network. And interface to the Mass Memory is via the MM bus.

The IC network is triply redundant and consists of three layers of a circuit switched node network. Three of these nodes, one from each layer, interface with the multiprocessor. On the multiprocessor side one processor triad interfaces with the triplex IC network (also referred to as the IC bus) at any given time. For data transmission, each member of this triad (called the IC triad) transmits on one bus. For receiving data, each IC triad processor listens to all three IC buses, deskews the data and votes on the three copies in hardware. Any bus disagreements trigger IC Error Latches in the processor-bus interface hardware. The identity of the disagreeing bus is stored in the Error Latch. Before performing bit by bit voting on incoming data, each serial data stream is checked for compliance with the bus protocol. Any deviations are recorded in error registers. The error latches and the protocol error registers can be read by a processor triad on the multiprocessor bus. If the data source GPC is not redundant, then voting circuitry is bypassed. Simplex source congruency is performed on the incoming data by writing it to shared memory and reading it back again. Since the IC bus is a contention bus, the FTMP contends for access to the bus with the other GPCs using a distributed arbitration algorithm known as the Laning Poll. Details of the Laning Poll scheme are described in Section 6.1. One of the FTMP triads that is enabled on the IC bus participates in this poll.

The I/O bus is a single layer, circuit switched network. At any given time, one processor triad (called the IOP triad) is assigned to interface with the I/O network. The IOP triad operates in a fashion similar to the IC triad. However, there is one major difference in that only one IOP processor transmits on the I/O bus at any given time. All three I/O processors listen to the I/O bus and perform source congruency on all incoming data by writing it to the shared memory. Since the I/O bus is a contention bus, they all participate in the bus arbitration by each member listening to the bus but only one of them actually transmitting on the bus. Laning Poll is used to arbitrate I/O bus conflicts.

Interface to the mass memory triplex bus is very much similar to the IC bus interface.

The transmissions from a processor on the IC, I/O and MM buses are gated through enabling gates. The purpose of the enabling gates is to protect the buses from runaway processors that can not otherwise be turned off. The enabling gates allow a processor to transmit only if a majority of the processors agree to do so. Each processor sends an enabling signal to every other processor. These enabling signals are voted upon to create a master enable signal in each LRU. If the multiprocessor is built with some slots initially unoccupied, the master enable signal is created by voting enable signals from only those processors which are present. Each LRU also produces a presence signal to indicate whether it is populated. The presence signal is also helpful in writing FDIR software and therefore should be made available as part of some LRU specific control register.

3.1.4.2.3 Microprocessor Level Architecture The AIPS microprocessor should have features that support the overall AIPS goals and requirements. From an architecture viewpoint the desirable features are the ones that support growth, fault tolerance, and high software reliability. These are described in the following.

3.1.4.2.3.1 Virtual Memory The microprocessor should support virtual memory.

There are three reasons for this. The first reason is that virtual memory capability provides simple and straightforward means of providing memory protection. Access authority of a process for a page of physical memory can be read/write, read-only or neither. This can be done by changing bits in the access field of the mapping register or mapping memory by the operating system. By limiting access of processes solely to program and data memory needed by that process one can contain the effects of software faults that result from out of range memory accesses by the faulty software module. Although virtual memory is not essential for providing memory protection, it is nevertheless a very convenient way of implementing it.

The second reason for having virtual memory is that it conveniently maps virtual memory into physical memory or secondary storage into primary storage. In the multiprocessor, the primary storage is the local cache (RAM and PROM in each processor) while the secondary storage is the shared memory. Having virtual memory simplifies the task of determining whether the required program/data is present in local cache or whether it should be loaded from the shared memory. By treating shared memory as virtual storage and cache memory as physical storage, the mapping registers can be set up to indicate whether a particular page of shared memory is present in the cache. If it is present the mapping register would point to its cache address and also indicate the access rights for the current process.

The third reason is that the virtual memory capability can be exploited to migrate tasks from one processing site to another. By allocating a unique address space to each such task in the system all the migratable tasks can be linked with local tasks for each computer where they might ever be migrated. However, the migratable tasks need not occupy physical memory in each target computer. They exist only in the virtual memory. When a task needs to be migrated to a new processing site, then the physical memory of the target computer would be loaded with the program/data of the migrating task and its virtual memory mapping registers altered to reflect the new contents of the physical memory.

Any one of the three uses cited above for virtual memory capability can be supported by other means. For example, memory protection can be provided by segmenting physical memory and attaching access rights to each segment. However, virtual memory makes it very convenient and straightforward to support all of these requirements with very little additional hardware and software.

3.1.4.2.3.2 **Privileged Mode** The microprocessor should support two modes of operation.

The two processor modes are usually referred to as the user mode and the supervisor mode. The supervisor mode is the privileged mode in which all operations are legal.. The user mode is more restricted and execution of certain instructions, called privileged instructions, is disallowed. Executing privileged instructions in the user mode causes a processor hardware interrupt.

This feature is necessary for software fault containment. As an example of this, consider the writing of memory mapping registers. If every process has the capability to alter the memory map, then it would not be possible to provide memory protection. Therefore, it is necessary that only the operating system, for example, be able to modify the memory map and the access rights of processes to various memory segments. Similarly, the ability to change system configuration control registers should only be restricted to the operating system and applications processes should not be able to alter the system configuration.

If the microprocessor does not support privileged mode of operation, one would have to rely heavily on static and dynamic analysis of programs to prevent unauthorized execution of certain instructions. Although static analysis of user programs could catch some unauthorized accesses, it isn't possible to prove that none would occur at run time.

Privileged mode operation is one more means of containing damage inflicted by faulty software. One would still do all the static and dynamic testing of software that one can afford in order to improve the reliability of each software module but building a lot of "fire walls" between software modules will assure that unreliability of one module does not affect the reliability of other pieces of software.

3.1.4.2.3.3 HOL Support The microprocessor should be able to support High Order Language(s) efficiently.

There are two reasons for this. First, there are considerable advantages of using HOLs to write software as opposed to using assembler level languages. These advantages are too numerous and already well known and will not be discussed here. Suffice it to say that since HOL use is mandated by AIPS requirements it would be prudent to select a microprocessor that supports the selected HOL efficiently.

Since Ada has been selected as the AIPS programming language, the selected microprocessor should ideally support certain features of this language efficiently. The most important feature of Ada from the fault tolerance viewpoint is the constraints on variables and the range checking. Currently, there are no microprocessors that support this more efficiently than others. That is, none of the commercial microprocessors has built-in hardware that can expedite range checking.

3.1.4.2.3.4 Memory Address Space The microprocessor should support "large" directly addressable memory space.

There are several reasons for this. The most obvious reason from AIPS program viewpoint is to meet the growth requirement. One should be able to "grow" the AIPS architecture in the memory dimension. It is easy to add physical memory to a processing site if it can be directly addressed by the microprocessor. If the additional memory is outside the directly addressable range of the processor, one has to add hardware such as a memory management unit to avoid this limitation.

What is "large" of course depends on the application. For AIPS, several megabytes per processing site would not be an unreasonable requirement. That is, the microprocessor address field should be at least 22 bits or longer.

3.1.4.2.3.5 Testability The microprocessor should have built-in features or support peripheral hardware that aids in the system testability.

Ideally, there should be a trace capability built into the microprocessor. This would capture a trace of the processor activity and could be used to trace back processor states from a given trigger event. None of the commercially available microprocessors support this feature in real-time. Currently, this is accomplished by work station emulators which observe system activity on the address, data, and control lines. The emulators and analyzers are continually evolving and becoming more and more sophisticated. It would be hard to build anything in the processor or around it to match these capabilities. The main drawback of emulators and analyzers is that they are useful only in the hardware/software development environment. They obviously can not be used in real time or even off line in an operational environment for testing and maintenance.

Other features that might enhance system testability are the built-in hardware breakpoint registers, capability to inject faults, and so on. These features, however, are not likely to be available in any commercial microprocessors but will be built into the surrounding hardware as discussed in the testability section.

3.1.4.2.3.6 Performance The microprocessor should have adequate performance to support the AIPS requirements of throughput, dynamic range, and precision.

Performance can be measured in several different ways. One measure is the average instruction cycle time. Of course not all microprocessors have equally powerful instructions and the instruction weighting factor can also make a difference in comparison.

Other factors that affect performance are the data path width and the floating point hardware. At the least, the data path should be 16 bits wide. A 32-bit microprocessor would obviously be able to do more in the same time and also provide more precision and dynamic range.

Having hardware floating point arithmetic capability also enhances the overall processor performance. The next best thing is a floating point coprocessor. Software floating point packages in lieu of the hardware may be acceptable from a performance viewpoint. But they add complexity to software development process since such a package would most likely be developed and then integrated with a commercial HOL compiler.

The performance criterion for microprocessor selection should take all of these factors in account.

3.1.4.2.4 IC and I/O Networks The circuit switched nodes of the IC and the I/O networks are identical. For the proof-of-concept system each node will have five identical ports. The node will interface with other nodes, GPCs, and I/O subscribers (displays etc.) through these ports. All ports will be identical in terms of their ability to interface with any of the aforementioned entities.

The Intercomputer communication network for the proof-of-concept configuration consists of three identical layers of a circuit switched network. Each layer consists of five nodes. Each node services one GPC. Although it is possible to service several GPCs from one node provided the node has enough ports to do this, this is not the case for the POC configuration.

The three layers of the IC network are totally independent and are not cross-strapped to each other. The initial no-fault configuration of the three layers is identical although it does not have to be so. That is, after a link failure in one layer the virtual bus configuration of that layer would change as the network is reconfigured around the failed link. The other two layers do not have to be reconfigured to make their virtual bus path identical to the third one. The fault detection, isolation, and reconfiguration of the IC network are the responsibility of the Global Computer. Nodes keep track of any transmission errors which are protocol related and inform the Global of these errors when queried by the Global. This error data is analyzed by the Global to determine source of transient faults on the network. The nodes also respond to status queries with the status of the node and the ports. Other than this, the nodes are totally passive circuit switching devices. They listen for node reconfiguration commands from all ports whether or not that port is active. Valid reconfiguration commands must be preceded by a Gateman code. Reconfiguration commands are addressed to individual nodes although they are heard by all nodes.

The principles of operation of the I/O network are same as that of the IC network. The I/O network configuration for the POC system consists of a single layer of 16 nodes. This network can be configured either as a single global I/O network or as several regional I/O networks. Both configurations will be used in the POC to demonstrate the global and regional I/O bus features of the AIPS system. System displays and controls would be attached to the I/O nodes.

3.1.4.2.5 **Mass Memory** The AIPS proof-of-concept mass memory requirements can be satisfied by Winchester disks, magnetic bubble memory, or Electrically Erasable ROM (EEROM) semiconductor memory. The choice will be made during preliminary design. The mass memory redundancy scheme will depend on this choice. If the decision is to use the disk or the magnetic bubble memory the memory will be replicated. Each copy will interface to the triplex mass memory bus through its own bus interface circuitry. The interface will be responsible for receiving redundant commands, address, and data from GPCs and performing deskewing, voting, and fault detection on the incoming data. Voted data will be stored in the memory. Each interface would also respond to the memory read requests on one of the three mass memory buses. If the memory is implemented as EEROM, the fault tolerance will be provided through encoding rather than triplication. A candidate encoding scheme is described in Section 6.2.

The mass memory interfaces will restrict access to mass memory by simplex GPCs to read-only operations. A 'Mass Memory bus hog' capability will also be provided in support of semaphores or locks associated with shared data in the mass memory. A GPC will be able to retain control of the MM bus, after gaining access to it, for multiple memory transactions. One can read a memory location, modify it, and write it back as a single atomic operation.

3.1.5 AIPS POC System Operating Environment.

3.1.5.1 **Overview.** The AIPS architecture, defined in this specification, provides the flexible and adaptive mechanisms needed for implementing system configurations for the diverse range of NASA applications. The AIPS multiapplication requirement necessitates that the architecture embody a broad range of concepts and implementation flexibility. Configurations for the multiple applications may be quite different from one another, using various of the system building blocks and concepts, and even within an application there may be valid considerations for more than one configuration.

The Proof-of-Concept (POC) system includes those AIPS concepts and system components which require or which are required for concept proof by formal test. The POC system serves three fundamental purposes.

(1) The development and "proof" of AIPS architectural concepts. The concepts include, for example, function migration, graded redundancy, and fault tolerant intercomputer communication.

(2) The development of system components necessary to implement the architectural concepts. This includes hardware, software, and control algorithms.

(3) Provides a facility, when combined with the operating environment, for the development and test of complimentary concepts, such as fault tolerant software.

The Proof-of-Concept system in its operating environment must be able to demonstrate the various AIPS architectural concepts sufficient for concept evaluation. The AIPS system operating environment is provided by the AIPS Integration and Evaluation Facility.

The System Integration and Evaluation goal is to verify that functional requirements have been met and to provide a broad characterization of the AIPS Proof-of-Concept system. The integration activity will be directed toward the requirements and the evaluation activity, while being guided by the requirements, will be exploratory in nature. System evaluation measurements will stress as many of the performance parameters as are meaningful in the context of the Proof-of-Concept system. The following specific types of evaluation tests will be performed:

(1) <u>Performance</u>

(2) <u>Failure Modes and Effects</u>

An integrated approach will be taken in performing the above tests types. Fault tests, for example, will be performed with the system stressed to determine system behavior during fault handling and reconfiguration. The following system services will be tested with varying processing and communication loads on the processors, IC network, and the I/O network(s). The list is not exhaustive and represents the minimum scope of the evaluation tests. The test details will be developed and documented in the AIPS test plan.

(1) Interfunction Communication (Inter/Intra Processor)

- System Behavior with Loaded Intercomputer Network
- Message Transport Delays
- Communication Overheads

(2) System Time Management

(3) Function Migration

- Reconfiguration Time
- Effect on Nonmigrating Functions

(4) System Fault Tolerance

- Simulate Random and Correlated Faults
- Verify/Update FMEA Models
- Detection, Isolation and Reconfiguration

(5) Global and Regional Sensor, Effector and Function Communication

- Transport Lag
- Communication Overheads

(6) Operating System Overheads

- Computer Redundancy Management
- Sensor/Effector I/O
- Interchannel Communication (for source congruency)
- Operating System Context Switch

3.1.5.2 Operating Environment. Figure 8 on page 38 shows a schematic of the POC system and its laboratory environment. The figure shows the five POC processing sites and a typical configuration of the intercomputer network (IC) and its nodes. Also indicated are direct connections between the host computer and each of the processors, the global I/O bus, and examples of regional and dedicated I/O busses. The mass memory and the memory busses are shown, and the power distribution system is shown without its connections. Not shown in the figure are fault injectors, which can be attached to any of the POC elements in which faults are to be created.

In integration, evaluation, and applications testing, the host computer is the primary test driver. The test operator will communicate with the host through an interactive control and display terminal, and the host will be equipped with appropriate printers, plotters, and mass storage devices, as well as suitable display software.

The host computer will be able to obtain and download POC software into the POC system. It will also be capable of the following functions:

- start and stop system, processors and channels within processors
- examine/modify processor memory, registers, or I/O ports
- examine/modify POC mass memory
- monitor IC network, including nodes
- monitor I/O bus traffic
- symbolic debugging

These functions will require appropriate software in the host, as well as corresponding hardware and software in the POC.

Also included in the facility is a subsystem workstation. In some tests, the subsystem workstation (or more than one) may be connected to the POC to provide very detailed monitoring of a specific processor, complementing the capabilities of the host/interface system. The subsystem work-

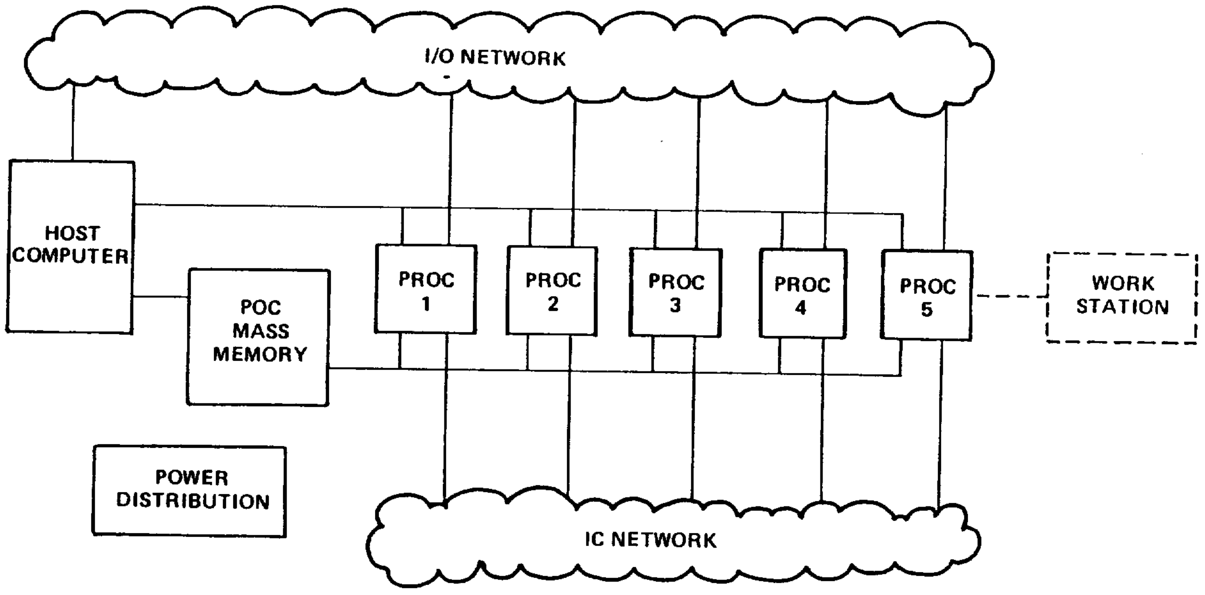

Figure 8. Proof-of-Concept System Operating Environment

station is connected to the POC in place of a processor and emulates that processor. While the emulation is transparent to the POC, it provides the capability to monitor and control the emulated processor. Typical functions supported by the subsystem workstation are:

- breakpoints
- start and stop processor
- tracing
- single stepping
- examine/modify processing site internal status
- symbolic debugging
- logic analysis
- statistical analysis

In addition, the subsystem workstation can function on its own as a general purpose computer, and is normally equipped with tools for software development (compilers, assemblers, etc.). In the POC environment, however, the software development will take place on a larger timesharing computer, which may also be used as the POC facility host, or at least will be linked to the host for data transfer.

3.2 Characteristics.

3.2.1 Performance Goals. The performance goals of the AIPS Proof-of-Concept System are derived from the AIPS requirements document, CSDL Report AIPS-83-50, and performance analyses accomplished during phase 1.

3.2.1.1 System Time. The global computer will broadcast time to the other GPCs over the IC network. This broadcast message will occur at 1 Hz. The nonglobal computers will align their internal clocks to this broadcast time. The related performance issues are:

(1) Time value granularity. The 'least significant bit' or granularity of the time value will be 100 microseconds or less.

(2) The total time skew between any two GPCs due to all sources will be no more than 1 millisecond.

(3) these performance estimates relate to error free conditions.

3.2.1.2 Function Migration. The performance goal is to minimize the time that a critical function is suspended due to migration. This minimum time applies to the condition where the function need be suspended only long enough to assure that the data transferred to the new site is consistent. It includes the time required to transfer the data via the IC bus and the time required for the new site to restart the function. The goal for the total time of function suspension is 50 milliseconds or less.

3.2.1.3 Fault Detection and Reconfiguration Time. For the FTMP the FDIR performance goals are:

Detection - 100 milliseconds
Identification - 50 milliseconds
Reconfiguration - 50 milliseconds

For the FTP similar times are applicable for detection and identification. Reconfiguration takes place only in the transition from duplex to simplex; this performance goal is 50 milliseconds.

3.2.1.4 Computer Redundancy Management Overhead. For the FTP and FTMP the FDIR overhead should be no more than 5% of processor throughput.

3.2.1.5 Interfunction Communication. The performance goals for interfunction communication are dependant on whether or not the communicating functions are in the same or different GPCs.

The goal is .5 milliseconds or less to send data from one function to another in the same GPC.

The goal is 2.5 milliseconds or less to send data from one function to another in a different GPC.

3.2.1.6 Sensor/Effector I/O. The performance goal for the cost of initiating and completing an I/O transaction on any of the I/O buses is .3 milliseconds. This is the fixed cost per transaction and does not include any bus transit time.

3.2.1.7 Interchannel Communication Time to Support Source Congruency. The goal for both the uniprocessor and the multiprocessor is 10 microseconds or less per 16 bit word.

3.2.1.8 Operating System Context Switch Time. The goal for both the uniprocessor and the multiprocessor is 150 microseconds or less.

3.2.1.9 Time for Intercomputer Messages. The goal, assuming no queueing delay, is 1.6 milliseconds or less per message. This includes the time for both the sender and receiver.

3.2.1.10 Total System Software Overhead. The goal is to provide at least 75% of the CPU throughput to application software. System Software should consume no more than 25%.

3.2.2 Physical Characteristics. The proof-of-concept system will be constructed with separate enclosures for each processing site. Basic packaging will be dual-inline components on wire wrap cards mounted in cages and cabinets. Site power conditioning and required cooling will be provided within the enclosure.

The packaging will be adequate for the laboratory environment and for shipment in appropriate containers.

3.2.3 Design Objectives The design objectives described in the following paragraphs shall be addressed as part of the system design and implementation. The design objectives are highly correlated and cannot be incorporated into a system implementation without consideration of the interactions. The architectural flexibility will allow the optimum balance of design objectives which will be a function of the relative importance of the various design objectives to the application program.

3.2.3.1 Physical Dispersion The AIPS elements shall be capable of physical dispersion for reasons of environmental protection, damage protection, or proximity to sensors and effectors. Physical dispersion should include the dispersion of redundant elements of a redundant set in order to provide damage tolerance.

3.2.3.2 Function Distribution The AIPS POC system shall support the distribution of application functions among the processing elements. Function distribution shall be allowed to change, via function migration, in response to system requirements for resource/function reallocation or in response to component failures. The allowed allocation of function to processor resource shall include the static no reallocation case.

3.2.3.3 Maintainability. Maintainability will be achieved by a number of different architectural and implementation attributes;

(1) Testability and fault isolation
- Modularity - hardware and software
- Embedded test features - hardware and software
- Error/Failure Logging

(2) Commonality
(3) Accessibility

3.2.3.4 Reliability. The AIPS architecture shall provide for the achievement of reliability levels commensurate with a range of applications and their respective maintenance strategies (e.g. AIPS configurations with functions requiring failure probabilities per mission range from 10^{-9} for a 10 hour mission with no repair to a 10^{-2} for a 20 year mission with repair).

The AIPS system reliability for a particular application may be tailored to match the specific application functions to be implemented. The AIPS architectural concepts and system components allow this to be performed through choices which consider other system attributes such as affordability. Reliability will be achieved through architectural and implementation attributes such as;

(1) Graded redundancy - tailored to application requirements.
(2) Function Migration - provides graceful degradation.
(3) Implementing technology
(4) Fault and damage tolerance
(5) Maintainability

3.2.3.5 Availability The AIPS architecture shall provide for the achievement of high levels of operational availability for those applications such as commercial and military aircraft that require a high rate of mission executions. The AIPS shall use attributes such as;

(1) Fault and damage tolerance
(2) Maintainability
(3) Testability
(4) Function migration
(5) Graded redundancy

to allow timely repair after a mission, to allow initiation of a mission with failed components, and to allow delay of maintenance until a more convenient time or place.

3.2.3.6 Adaptability. The AIPS architecture shall minimize sensitivity to changes in functional requirements, performance goals, and levels of desired reliability and/or fault tolerance. It shall be possible to physically or functionally add to an implementation of the AIPS architecture while minimizing the effect on existent functions with which the added function has no communication.

Adaptability will be achieved by architectural and implementation attributes such as;

(1) Modularity
(2) System Expandability
(3) Component Commonality
(4) Reconfigurability

3.2.4 Testability

3.2.4.1 Testability Concept. The testability concept for the AIPS architecture embodies three distinct test environments although each test feature, listed in the following sections, may provide support for more than one. The three test environments are.

(1) Operational support for fault detection, isolation, and recovery.
(2) Operational support for system maintenance
(3) Development Support

This section applies to major system test features that are primarily required for development test and does not address the full range of fault detection required for operational fault detection and isolation.

3.2.4.2 System Test Features. The AIPS system shall incorporate features to enhance testability.

3.2.4.2.1 Test Features.

3.2.4.2.1.1 System Fault Logging. The software shall record the occurrence of all detected hardware and software faults. The record shall contain the following information about each fault.

(1) Detection time (real-time clock)
(2) Fault source and identification time (if identified)
(3) Reconfiguration time

If fault source not identified - then provide list of suspects. The global computer shall maintain the system faults log.

3.2.4.2.1.2 Resource Utilization. The local operating systems shall monitor and record the usage duty cycle of their respective hardware resources. The primary purpose for this requirement is the analysis of system performance and the identification of throughput bottlenecks.

3.2.4.2.1.3 System Configuration Trace. The global computer operating system will be capable of recording a chronological history of system configuration. Reconfiguration will occur for a limited set of reasons. These reasons will be characterized and a code for each defined. For each reconfiguration event the reason code and reconfiguration time will be recorded with the reconfiguration data. This record will contain, for example, the following types of information.

(1) Intercomputer Network configuration history
(2) Input/Output Network configuration history
(3) Function Assignment history

3.2.4.2.1.4 System Halt The system processing components shall provide the ability to "halt" the individual processing components individually or collectively. This shall be possible without the loss of data currently being transferred between processing sites. The "halt" shall be capable of being disabled.

The halt will be a result of internal address and/or data compares at a processing site or by external command from test equipment.

- internal address compare - The halt will occur based upon the fetch of address and/or data at a processing site. The address/data may be specified by an external device.
- external command - An interrupt may be requested by external devices. Two such types of external devices will be allowed; a processor at a different processing site may cause a halt or an external test device may cause a halt. This feature will be used to implement the system halt. The halt may be inhibited at the local processor site and/or by external test equipment.

3.2.4.2.1.5 System Watchdog Timer. The system watchdog timer is implemented within alternate global computers. It is referred to as the 'heartbeat monitor' whereby the alternate global assumes the roll of the global if it does not receive a periodic poll message for a predetermined number of poll cycle intervals.

3.2.4.2.2 Processor Site Test Features

3.2.4.2.2.1 Fault Logging The software shall record the occurrence of all detected hardware and software faults. The record shall contain the following information about each fault.

(1) Detection time (real-time clock)
(2) Fault source and identification time (if identified)
(3) Reconfiguration time

If fault source not identified - then provide list of suspects. The local computer shall maintain the local fault log.

3.2.4.2.2.2 Operating System Entry Trace. Each operating system will be capable of recording a chronological history of invoked entry points. The history trace will identify the operating system entry point, the time of the entry, and an identification of the invoking process. Certain software and hardware errors will cause the recording of the trace to stop. In addition, the operator will be able to start and stop the trace. Each GPCs' history trace will be separately controlled.

3.2.4.2.2.3 Watchdog Timer. There are two distinct implementations of the watchdog timer at the processing sites.

- synchronous - The synchronous watchdog timer will be implemented through the use of a program interval timer. The interval timer is set to the maximum allowable time for a process to complete. At the end of the process the timer is reset for the next process to be scheduled. If the process does not complete within the set interval an interrupt occurs notifying the operating system. The synchronous watchdog timer is intended to detect software process errant behavior manifested by longer than expected operating time.
- asynchronous - The asynchronous watchdog timer provides additional fault coverage. For example, failures in the hardware or software that would disable software implemented fault detection may be

detected by an asynchronous watchdog timer. The device will report a channel to its partner channels and disable its own output circuitry. The asynchronous watchdog timer must be serviced by the system software periodically in order to prevent the failure indication. The asynchronous watchdog timer will not use the processing site oscillators or fault tolerant clock for operation and in the unpowered state will indicate "fail".

3.2.4.2.2.4 Processor Halt Each processor at each processing site will have the capability to be "halted" based upon internal and external events. The "halt" will be implemented through the use of a program interrupt which when not masked will cause the execution of a test routine. The test routine will allow an external device to request internal data and to provide data for use by the test routine or other software resident at the processing site.

- internal address compare - The halt will occur based upon the fetch of address or data from a specified address. The address may be specified by an external device.
- external command - An interrupt may be requested by external devices. Two such types of external devices will be allowed; a processor at a different processing site may cause a halt or an external test device may cause a halt. This feature will be used to implement the system halt. The halt may be inhibited at the local processor-site and/or by external test equipment.

Access to processor internal state will be via the "halt" actuated software routine or other software routines accessible to external test and monitor equipment.

3.2.4.2.2.5 Simulated Fault Capability. In order to verify that the fault tolerance features of the system are functioning properly, it will be necessary to simulate the incorrect behavior of certain system components. To that end, it will be possible to add software that simulates the incorrect behavior of a system element or its components. In particular, it will be possible to simulate a fault in any single element of a redundant complement, for example, a single member of an FTMP processor triad.

3.2.4.3 Testability Matrix. The test matrix depicted in Figure 9 on page 45 shows the correlation between the system and processing site test features and the matching test types.

3.2.5 Environmental Conditions. The operating environment shall be as follows.

3.2.5.1 Ambient Temperature. The upper and lower temperature operating limits for the deliverable POC systems will be 60 degrees Fahrenheit and 90 degrees Fahrenheit respectively.

3.2.5.2 Humidity. The upper and lower relative humidity operating limits for the deliverable POC systems will be 30% and 95% respectively.

	(1) FDIR	(2) MAINTENANCE	(3) DEVELOPMENT
SYSTEM TEST FEATURES			
System Fault Logging	X	X	X
Resource Utilization		X	X
System Configuration Trace		X	X
System Halt			X
System Watchdog Timer	X		X
PROCESSOR SITE TEST FEATURES			
Fault Logging	X	X	X
Operating System Entry Trace	X	X	X
Watchdog Timer	X		X
Processor Halt			X
Simulated Fault Capability			X

Figure 9. Testability Matrix

3.2.5.3 Electrical Power. The electrical power requirements will be TBS.

3.2.5.4 General. The POC system is intended to be operated in the protective environment of a development laboratory. It is not intended to be used in an environment where shock, vibration, temperature extremes, etc. are to be encountered.

3.3 Design and Construction

3.3.1 Parts, Materials, and Processes The POC system elements shall be designed using commercially available integrated circuits, microprocessors and other piece parts. Programmable logic arrays may be used where appropriate. It is not required that custom integrated circuits be used or that custom components be developed for the proof-of-concept system.

Component selection will consider the availability of Mil-qualified or qualifiable versions of the parts to be used. Designs will consider the parameters of full range parts, although the proof-of-concept system will use commercial and industrial grade devices.

3.3.2 Electromagnetic Radiation The EMI emissions and susceptibility will conform to that nominally associated with development laboratory grade systems.

3.3.3 Workmanship The proof-of-concept system must operate satisfactorily in the test laboratory environment and endure changes, maintenance, reconfiguration, and fault injection (e.g. repeated component removals and installations). Workmanship standards suitable for these laboratory conditions will be implemented.

The proof-of-concept system design will use proven packaging techniques. The first build of a component will be hand wired; subsequent builds, including the deliverable systems, will be built by automatic wiring machine.

3.3.4 Computer Programming Standards The proof-of-concept software will be produced using programming standards in conformance with the CSDL 'Software Development Policies and Guidelines'. It is intended to use the Ada Programming Language and an implementation of its associated Ada Programming Support Environment, including program design languages and configuration management tools to develop, manage and control the software.

3.4 Documentation This specification states the overall system description and technical requirements for the development of AIPS Proof-of-Concept (POC) systems. From these requirements, detailed technical requirements shall be derived for AIPS. Those requirements will form the basis for the detailed design of an AIPS POC system described in up to four sets of design specifications (AIPS hardware, AIPS software, test support elements, development support elements) as shown in the Figure 10 on page 47.

One set of design specifications will cover the detailed requirements and description of the major hardware elements of AIPS. The specifications will include sufficient information, along with referenced documentation, to completely describe each hardware element and its interfaces. This includes principles of operation, schematics, fabrication drawings, parts lists, and reference publications.

A second set of design specifications will cover the major software elements of the AIPS POC system, including the AIPS Evaluation and System Software, and Demonstration Applications Software. Detailed requirements for this software will be stated in separate requirements specifications. The designs which satisfy the detailed requirements will be described in the software design specifications. The design specifications will include characterizations of overall program flow and architecture, and detailed descriptions of the inputs, outputs, and logic of each of the computer programs. The software specifications will, in general, support the software development concepts presented in document CSDL-C-5526, "Software Development Policies and Guidelines".

The remaining sets of design specifications will cover the special, facilities related hardware and software features needed for AIPS Test Support and AIPS Development Support. Individual specifications of this type will include both the detailed requirements and design descriptions necessary

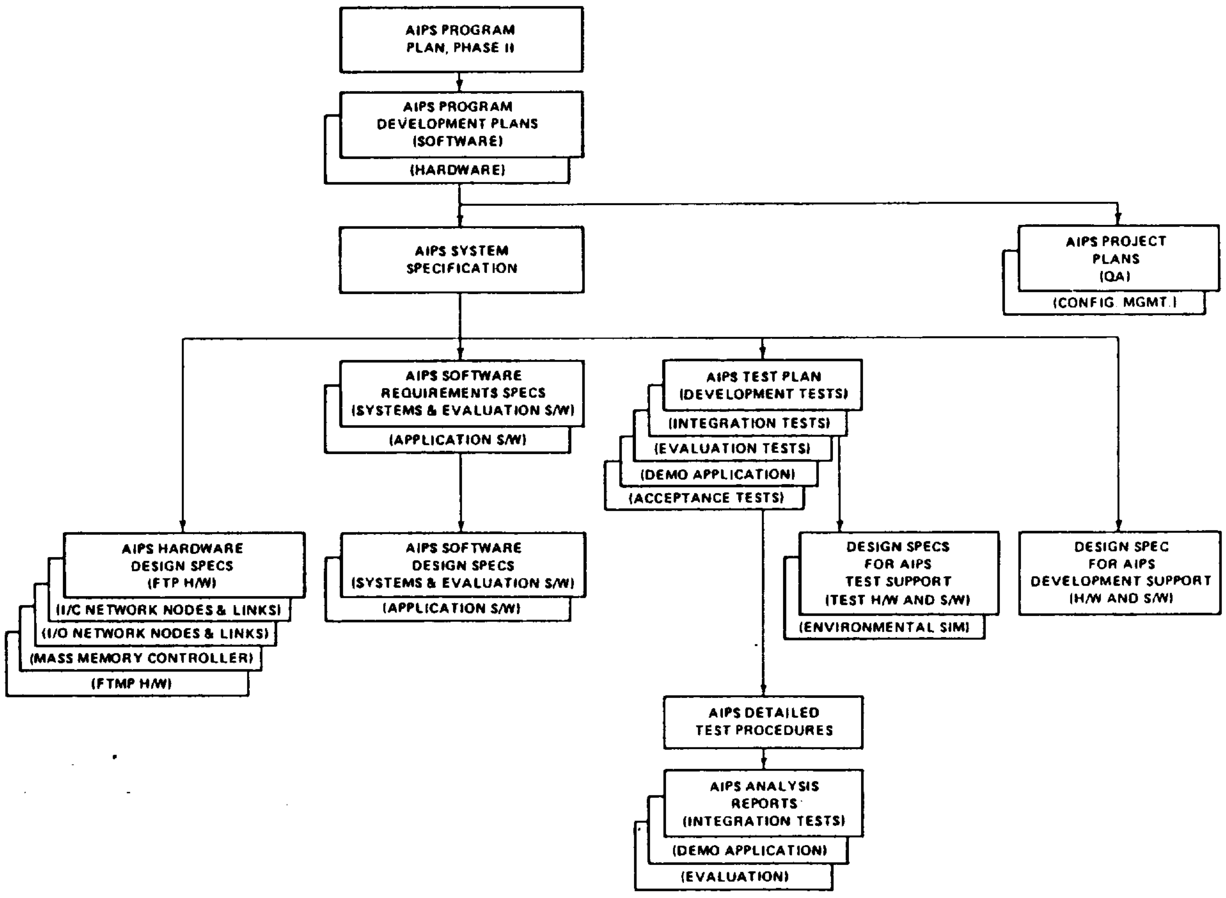
AIPS PROGRAM PLAN, PHASE II
AIPS PROGRAM DEVELOPMENT PLANS (SOFTWARE)
(HARDWARE)
AIPS SYSTEM SPECIFICATION
AIPS PROJECT PLANS (QA)
(CONFIG MGMT.)
AIPS SOFTWARE REQUIREMENTS SPECS (SYSTEMS & EVALUATION S/W)
(APPLICATION S/W)
AIPS TEST PLAN (DEVELOPMENT TESTS)
(INTEGRATION TESTS)
(EVALUATION TESTS)
(DEMO APPLICATION)
(ACCEPTANCE TESTS)
AIPS HARDWARE DESIGN SPECS (FTP H/W)
(I/C NETWORK NODES & LINKS)
(I/O NETWORK NODES & LINKS)
(MASS MEMORY CONTROLLER)
(FTMP H/W)
AIPS SOFTWARE DESIGN SPECS (SYSTEMS & EVALUATION S/W)
(APPLICATION S/W)
DESIGN SPECS FOR AIPS TEST SUPPORT (TEST H/W AND S/W)
(ENVIRONMENTAL SIM)
DESIGN SPEC FOR AIPS DEVELOPMENT SUPPORT (H/W AND S/W)
AIPS DETAILED TEST PROCEDURES
AIPS ANALYSIS REPORTS (INTEGRATION TESTS)
(DEMO APPLICATION)
(EVALUATION)

Figure 10. AIPS Documentation Tree

for the support of development and testing of the AIPS POC system at CSDL. The intent is to include modifications required by AIPS to software and hardware tools already available commercially or at CSDL, incorporating as much as possible by referencing existing documentation on these tools.

The methods in which testing of AIPS will be accomplished shall be described in an AIPS test plan. The plan will cover testing of both the hardware and software features to assure that the AIPS POC system meets the detailed and overall AIPS system requirements. The plan will also include evaluation and demonstration application tests which explore the achieved performance of AIPS.

The plans for the final integration and evaluation tests will be accompanied by test procedures sufficiently detailed to provide the capability to rerun tests with repeatable results. It is intended that these procedures will be suitable for acceptance tests of AIPS systems fabricated subsequent to the CSDL POC system.

The results of tests specified by the AIPS test plan, as well as conclusions from various AIPS studies will be summarized in a set of AIPS Analysis Reports as shown in the Documentation Tree.

3.5 Functional Area Characteristics

3.5.1 System Hardware. This section of the System Specification provides the system hardware requirements. It is intended that the subsequent sections specifying the individual hardware subsystem characteristics will provide the roots for subsequent design specifications.

3.5.1.1 Fault Tolerant Processor (FTP) The fault tolerant processor consists of processing elements interconnected via data exchange devices. An FTP can exist as a triple or dual processing element. Simplex processing elements will be an adaptation of an FTP channel. The basic configuration of a triplex FTP is shown in Figure 11 on page 49.
Each channel of an FTP is identical except the local I/O in each may be different. All execute identical software in synchronism. Each processing element of an FTP will contain a portion of the fault tolerant clock, a computational processor (CP), an input output processor (IOP) which contains interfaces to the intercomputer (IC) network, input output (I/O) network, and functions common to both the CP and IOP. All computational processors and input output processors of a fault tolerant processor are interconnected through data exchange registers.

In the proof of concept system each input/output processor of the FTP is connected to a seperate root node. Only one input/output processor of an FTP is active during an I/O transaction. Messages transmitted and received on the I/O network are handled in the same manner as processor unique I/O, that is each processor will output as if its transmitter were enabled whether is is or not. For an input the I/O processors will respond to an interrupt or flag and all will perform the input function, although only one will actually receive the data, and use the data exchange mechanism to share the data.

3.5.1.1.1 Computational Processor Each computational processor shall be comprised of the elements shown in Figure 12 on page 50.

3.5.1.1.1.1 Central Processor Unit The central processor unit (CPU) is the processing element for the computational processor. All data processing and manipulation is performed by the CPU. It will operate with a clock speed appropriate for the chosen processor and the processor shall function with virtual memory through a memory management unit. Memory accesses that are not resident within the local storage area will cause page faults. The processor shall be capable of suspending processing, reloading memory with the required data and restarting program execution without loss of continuity. Upon power up, all CPUs will be reset. This reset shall clear all working registers and flags. As part of the initialization the CPU shall reset all interfaces to a predetermined state.

3.5.1.1.1.2 Memory The memory module contains the local storage for the computational processor. It contains all program storage, constants and

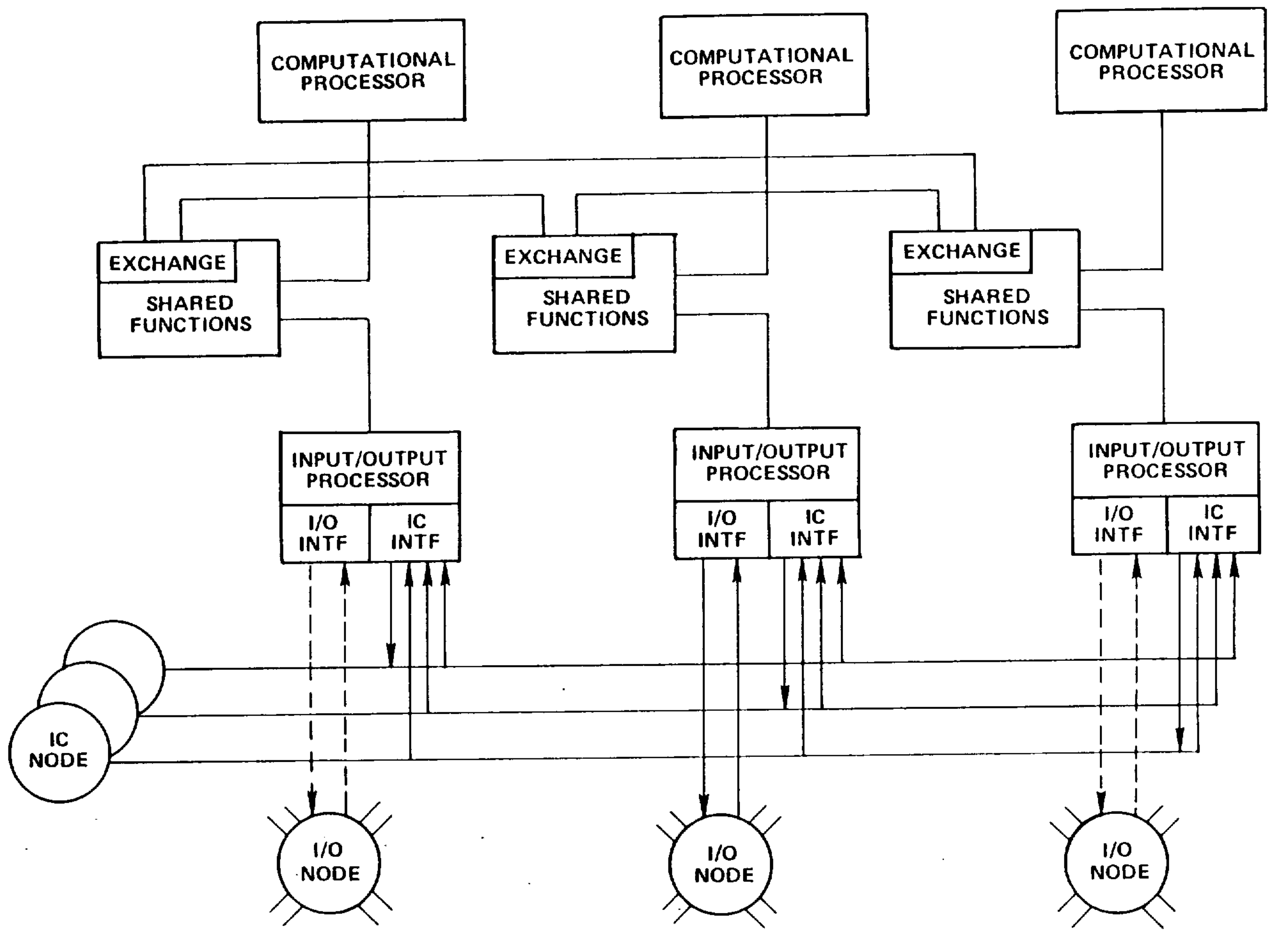

Figure 11. Fault Tolerant Processor Functional Diagram

variables necessary to perform the assigned tasks. Each memory shall be comprised of random access memory (RAM) and read only memory (ROM). The total of RAM and ROM is 256k bytes. The ratio of the two memory types will be determined during the preliminary design process.

(1) RAM

The RAM may be used for the storage of variables and data as well as functions migrated to this element. The RAM shall have an access time such that no more than two processor wait cycles are required for access.

(2) ROM

The ROM will contain the fixed operational programs, subroutines, constants, initialization and diagnostic programs. The ROM shall have an access time such that no more than two processor wait cycles are required for access.

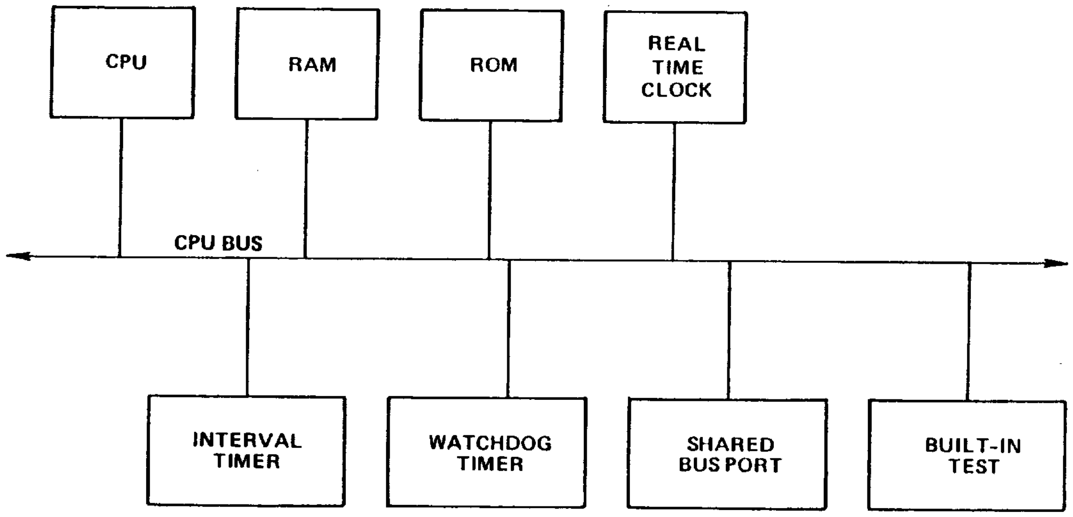

Figure 12. Computational Processor Functional Diagram

3.5.1.1.1.3 Timers The timers are used for keeping track of real time, timing tasks, and maintaining a check on the health of the hardware and software. Each timer module shall contain a real time clock, interval timer(s), and a watchdog timer.

(1) Real Time Clock

The real time clock shall be a 32 bit hardware register which can be preset to the system time under program control. Once set the real time clock shall increment every 50 to 100 microseconds, depending upon available timing. When the count reaches the maximum value it shall reset to zero at the next count cycle and continue to count up.

(2) Program Interval Timer(s)

The interval timer(s) shall be a 16 bit hardware register that can be preset to any value by the software. Once set the interval timer shall decrement to zero and continue. When the timer reaches zero an interrupt is requested. The interval timer(s) shall be capable of being read or written. The interval timer(s) shall have a resolution between 8 and 24 microseconds based upon available timing.

(3) Asynchronous Watchdog Timer

The asynchronous watchdog timer is a fault detection mechanism which provides an overall check on the health of the processing element. Under normal operating conditions the software must reset it periodically. This verifies that the system is functional. If not reset or a hardware failure prohibits service, the asynchronous watchdog timer shall reset the CPU. It shall be possible to inhibit the asynchronous watchdog timer when operating in the test mode. The asynchronous watchdog timer shall be independent of the processor oscillator.

3.5.1.1.1.4 Shared Bus Port Each processing element shall have a shared bus port. This port allows the processor to gain access to the devices that the CP and the IOP share. The items that they share are described in section 3.5.1.1.3. The shared bus port shall contain the contention logic and the bus drivers and receivers necessary to use the shared bus. A functional block diagram of a bus port is shown in Figure 13 on page 52. The processor requests use of the bus from the bus arbiter and if available it is free to perform a transfer on the bus, at the same time the other processor is prevented from using the bus. If the bus is in use the request will remain pending.

3.5.1.1.1.5 Built-in Test Features The system component shall have built-in test features to provide visibility into the system operation. The following features shall be included.

(1) Dual Port Memory

A dual port memory shall provide a communication path between the Proof-of-Concept system and the Integration and Evaluation Facility host computer. The facility host computer shall be capable of sequentially reading the data stored in all dual port memories in all elements of the system. Additionally, it shall be possible for the facility host computer to download programs and commands to the processing elements without use of the communication networks. This dual port memory will appear to the processing elements and the facility host computer as 2048 words of RAM memory.

(2) Bus Monitor

Each processing element shall have a nonintrusive bus monitoring capability. This monitoring capability should be capable of detecting specified events within the processor. This capability shall reside on an independent circuit board such that the feature may be installed or deleted as required. Detectable events shall be:

(a) Condition Compare

Each monitor shall have an address and data compare register which can be read or written by either the processor or through the external test device, such as the the facility host computer. The address compare register will be used to continually monitor the address bus. Whenever an address match occurs, several results shall be possible.

(i) A flag will be set indicating a match; This flag will be accessible by the facility host computer through the dual port memory.

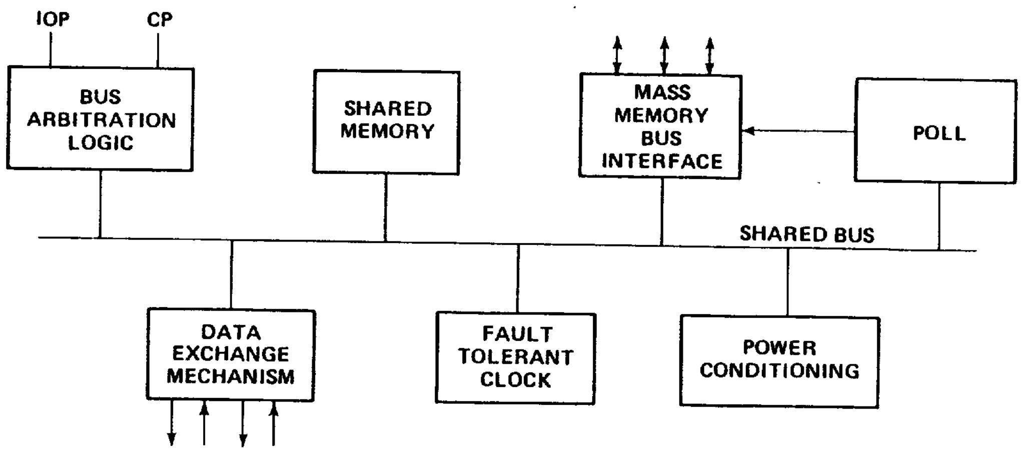

Figure 13. Shared Bus Port Functional Block Diagram

(ii) An interrupt will be provided and if enabled a "halt" of the processing element will occur.

(iii) A global halt signal shall be transmitted to other processing elements. (The results at other sites will depend upon the enable or disable of the signal.)

Each of these conditions shall be selectively enabled or disabled. In addition, response to an external halt signal, from another processing element, shall be selectively enabled or disabled. The address compare shall function on the address field of the processor and shall be capable of indicating a match within two fault tolerant clock cycles of its occurrence.

(b) Selected Exceptions

Each monitor shall be capable of displaying the occurrence of exceptions. A method of signalling an external test device whenever an exception occurs shall be provided. The exceptions implemented shall be documented in the hardware design specification.

(c) External Commands

Each processing element shall have the capability of accepting commands from external test devices. Commands that shall be implemented are: halt on condition; read memory; halt from external condition; start program execution.

(d) The test software shall have the capability to start or stop the real time clock, the interval timer(s), and the asynchronous watchdog timer.

3.5.1.1.2 Input Output Processor Each Input Output processor shall be comprised of the elements as shown in Figure 14 on page 53. The basic CPU, memory, timers, shared memory and dual port memory are identical to those in the computational processor. The IOP however contains some additional elements which are also defined.

3.5.1.1.2.1 Central Processor Unit The central processor unit (CPU) is the processing element for the computational processing element. All data processing and manipulation is performed by the CPU. It will operate with a clock speed appropriate for the chosen processor and the processor shall function with virtual memory through a memory management unit. Memory accesses that are not resident within the local storage area will cause page faults. The processor shall be capable of suspending processing, reloading memory with the required data and restarting program execution without loss of continuity. Upon power up, all CPUs will be reset. This reset shall clear all working registers and flags. As part of the initialization the CPU shall reset all interfaces to a predetermined state.

3.5.1.1.2.2 Memory The memory module contains the local storage for the computational processor. It contains all program storage, constants and variables necessary to perform the assigned tasks. Each memory shall be comprised of random access memory (RAM) and read only memory (ROM). The total of RAM and ROM is 128k bytes. The ratio of the two memory types will be determined during the preliminary design process.

(1) RAM

The RAM may be used for the storage of variables and data as well as functions migrated to this element. The RAM shall have an access time such that no more than two processor wait cycles are required for access.

(2) ROM

The ROM will contain the fixed operational programs, subroutines, constants, initialization and diagnostic programs. The ROM shall have an access time such that no more than two processor wait cycles are required for access.

3.5.1.1.2.3 Timers The timers are used for keeping track of real time, timing tasks, and maintaining a check on the health of the hardware and software. Each timer module shall contain a real time clock, interval timer(s), and a watchdog timer.

(1) Real Time Clock

The real time clock shall be a 32 bit hardware register which can be preset to the system time under program control. Once set the real time clock shall increment every 50 to 100 microseconds,

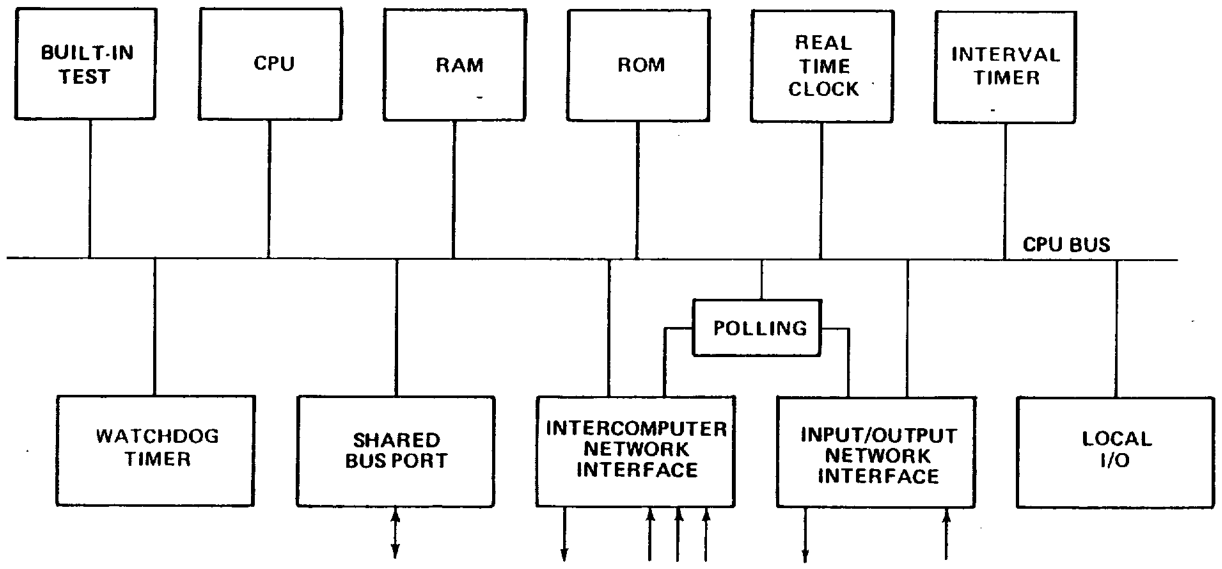

Figure 14. Input Output Processor Functional Diagram

depending upon available timing. When the count reaches the maximum value it shall reset to zero at the next count cycle and continue to count up.

(2) Program Interval Timer(s)

The interval timer(s) shall be a 16 bit hardware register that can be preset to any value by the software. Once set the interval timer shall decrement to zero and continue. When the timer reaches zero an interrupt is requested. The interval timer(s) shall be capable of being read or written. The interval timer(s) shall have a resolution between 8 and 24 microseconds based upon available timing.

(3) Asynchronous Watchdog Timer

The asynchronous watchdog timer is a fault detection mechanism which provides an overall check on the health of the processing element. Under normal operating conditions the software must reset it periodically. This verifies that the system is functional. If not reset or a hardware failure prohibits service, the watchdog timer shall reset the CPU. It shall be possible to inhibit the asynchronous watchdog timer when operating in the test mode. The asynchronous watchdog timer shall be independent of the processor oscillator.

3.5.1.1.2.4 Shared Bus Port Each processing element shall have a shared bus port. This port allows the processor to gain access to the devices that the CP and the IOP share. The items that they share are described in section 3.5.1.1.3. The shared bus port shall contain the contention logic and the bus drivers and receivers necessary to use the shared bus. A functional block diagram of a bus port is shown in Figure 13 on page 52. The processor requests use of the bus from the bus arbiter and if available it is free to perform a transfer on the bus, at the same time the other processor is prevented from using the bus. If the bus is in use the request will remain pending.

3.5.1.1.2.5 Intercomputer Network Interface. Each IOP shall contain an interface to the intercomputer (IC) network. This is shown in Figure 11 on page 49 as the triplex interface to the IC Node. A detail diagram of the major components of this interface is shown in Figure 15 on page 56 Each processing element listens to all three IC network layers, but transmits on only one network layer. The following describes the function of each interface element.

(1) Voter, Error Latch and Receiver

The receiver shall accept the serial data stream, verify compliance with the network protocol, convert to internal logic levels and timing, and check for transmission errors. If the data does not comply with the protocol, the receiver shall reject the transmission until a valid protocol synchronization is received. If a transmission error is detected, the event shall be stored in an error register. This error register shall store the errors until the CPU software reads and clears it.

In the case of the triplex intercomputer network there is a receiver for each of the three IC buses. The outputs from the three receivers are applied to each of the FTP IOP voters (one per redundant channel). The voter receives the triply redundant transmission from the Intercomputer network and does a bit by bit comparison of the transmission. If any of the bits disagree it shall output the value of the two out of three majority of that bit position. At the same time it shall set a flag in the error latch indicating on which line it perceived an error. The error latch flags are stored within the error register until acted upon by the software. When read by the software the voter or receive error registers shall be cleared.

In the case of a duplex intercomputer network (or triplex downmoded to duplex) the FTP voter will only have two inputs to function with. The output of the voter will always have an error in the nonexistent channel position. This error may be masked by the software. If either of the other channel positions indicates an error, then it will be an indication that there was a disagreement between the two data paths. With only two data copies the voter can detect an error but not isolate it. Isolation of the fault will depend upon the detection of errors at the receiver prior to the voter.

To support receipt of simplex data (including triplex/duplex downmoded to simplex) a bypass of the voter shall be allowed. In the simplex configuration, any data source received by the voter

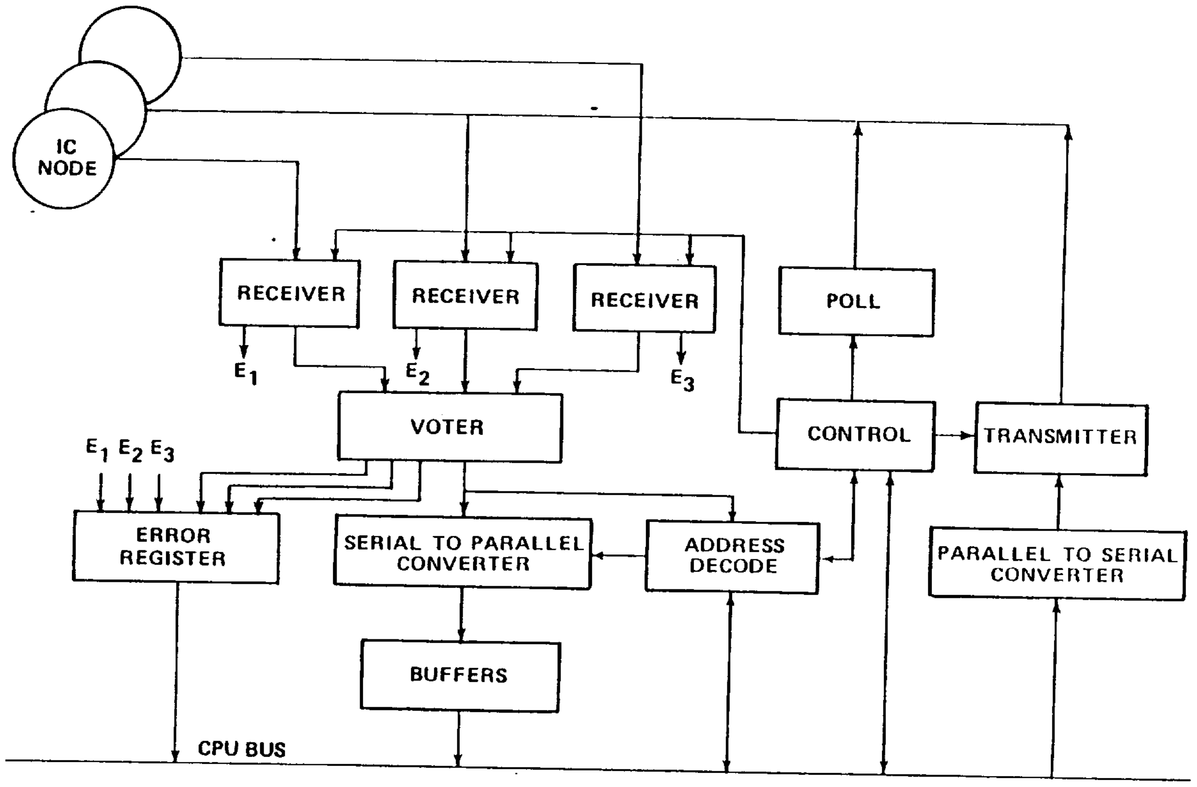

Figure 15. IC Network Interface Functional Diagram

may be selected. The bypass shall be constructed so as to insure that a failure causing the bypass mode is detectable.

(2) Data Conversion

The output of the voter is a serial bit stream that has had vote discrepancies masked. This output is then applied to a serial to parallel converter. The output of the serial to parallel converter can be read by the CPU through the buffers.

(3) Address Decode

Each device that is connected to the intercomputer bus shall have a broadcast address and a unique address for identification. All functional devices shall receive all messages that appear on the intercomputer network. It is the responsibility of the device to determine which messages are intended for it. This is the function of the address decoder. As each message is received, the address decoder will look at the address portion of the message and if it finds a match notify the CPU of a waiting message. The address decoder shall respond to broadcast messages as well as specific messages.

(4) Transmitter

Each channel of an FTP transmits on one intercomputer network layer. All IOP CPUs in the FTP load their Intercomputer transmit buffer with the data to be transmitted. The output of the buffers are serialized and applied to the transmitter, which encodes the message into the network protocol and if enabled transmits it.

The transmission duration shall be limited by the channel hardware and there shall be a delay in reenabling to protect against babbling mode faults.

(5) Control

The interface shall contain a control module that enables and disables functioning of the interface. It also provides the sequence control for the protocol encoder and decoder and is under software control of the CPU.

(6) Network Contention

Each transmission on the intercomputer network is preceded by contention for the network by those processing elements requiring access. The Laning poll technique shall be used for resolving contention. At the end of the previous transmission the protocol receiver and transmitter will be gated off and the contention logic enabled. Any processing element that requires the network will wait a prescribed time after normal transmission ends and start the poll. Any unit not entering the poll within a specified time of the start will be locked out of this poll sequence. Whenever an element observes the completion of the poll, even when it was not a participant, it will switch over to the data mode. This will allow all elements on the network to receive the data transmission, and will be performed totally in hardware so as not to burden the software. The polling is performed simultaneously on all three IC buses. When the poll is complete the FTP will do an exchange to verify that all elements have the same poll result.

3.5.1.1.2.6 Input/Output Network Interface The input/output network interface provides access to the I/O network. Each input/output processor has a direct interface to a dedicated network root node. However, only one input/output processor will be enabled to receive or transmit during an I/O transaction. When a message is received, a cross channel interrupt or flag will be generated so that all processors will perform the read. The data exchange mechanism will be used to perform source congruency. When an FTP desires to transmit the polling will be handled by the hardware in all IOP channels; however, the IOP attached to the root node addressed will perform the polling and subsequent transmission. The input resulting from the transmission, if any, will be received by the transmitting IOP and passed to the other IOP channels via the exchange mechanism. Since the I/O is simplex it follows that voting cannot be used to mask errors. Errors will be detected by the protocol checker in the enabled receiver. The connection of an FTP to the input output network is shown in Figure 11 on page 49. A block diagram of the input output network interface is shown in Figure 16 on page 58 and the function of the major elements follows.

(1) Receiver

The receiver shall accept the serial data stream, verify compliance with the network protocol, convert to internal logic levels and timing, and check for transmission errors. Transmission errors will be available to the software via a register.

(2) Data Conversion

The output of the receiver is converted from a serial bit stream to parallel words in the serial to parallel converter. These parallel words can now be read by the CPU through the buffers.

(3) Transmitter

Only one member of the FTP is enabled to transmit on the input output bus. All IOP CPUs in the FTP load their I/O transmit buffer with the data to be transmitted. The output of the buffers are serialized and applied to the transmitter, which encodes the message into the network protocol and if enabled transmits it.

The transmission length shall be limited by the channel hardware and there shall a delay in reenabling to protect against babbling mode faults.

(4) Control

The interface shall contain a control module that enables and disables functioning of the interface. The FTP software will determine which IOP is to be enabled for transmitting. The control module in the selected IOP will enable its transmitter with the other two disabled. It also provides the sequence control for the protocol encoder and decoder and is under software control of the CPU.

(5) Network Contention

Each transmission on the input/output network is preceded by contention for the network by those processing elements requiring access. The Laning poll technique shall be used for resolving contention. At the end of the previous transmission the protocol receiver and transmitter will be gated off and the contention logic enabled. Any processing element that requires the network will wait a prescribed time after normal transmission ends and start the poll. Any unit not entering the poll within a specified time of the start will be locked out of this poll sequence. Whenever an element observes the completion of the poll, even when it was not a participant, it will switch over to the data mode. This will allow all elements on the network to receive the data transmission, and will be performed totally in hardware so as not to burden the software.

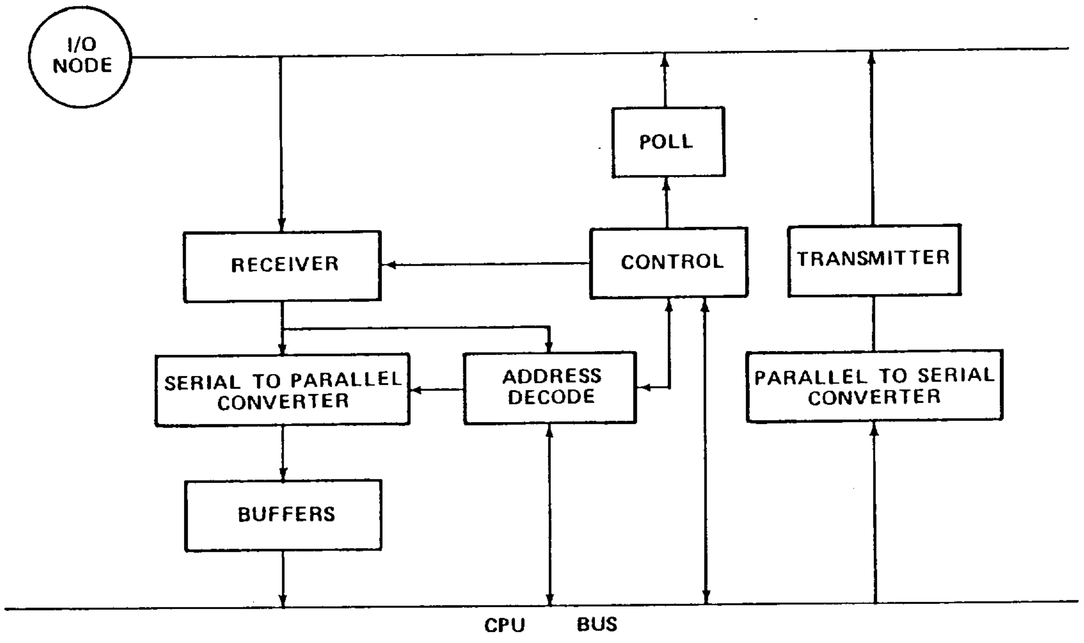

Figure 16. I/O Network Interface Functional Block Diagram

3.5.1.1.2.7 Local Input/Output Devices Each IOP shall be capable of accepting additional input/output mechanisms.

3.5.1.1.2.8 Built-in Test Features The system component shall have built-in test features to provide visibility into the system operation. The following features shall be included.

(1) Dual Port Memory

A dual port memory shall provide a communication path between the Proof-of-Concept system and the Integration and Evaluation Facility host computer. The facility host computer shall be capable of sequentially reading the data stored in all dual port memories in all elements of the system. Additionally, it shall be possible for the facility host computer to download programs and commands to the processing elements without use of the communication networks. This dual port memory will appear to the processing elements and the facility host computer as 2048 words of RAM memory.

(2) Bus Monitor

Each processing element shall have a nonintrusive bus monitoring capability. This monitoring capability should be capable of detecting specified events within the processor. This capability shall reside on an independent circuit board such that the feature may be installed or deleted as required. Detectable events shall be:

(a) Condition Compare

Each monitor shall have an address and data compare register which can be read or written by either the processor or through the external test device, such as the the facility host computer. The address compare register will be used to continually monitor the address bus. Whenever an address match occurs, several results shall be possible.

(i) A flag will be set indicating a match; This flag will be accessible by the facility host computer through the dual port memory.

(ii) An interrupt will be provided and if enabled a "halt" of the processing element will occur.

(iii) A global halt signal shall be transmitted to other processing elements. (The results at other sites will depend upon the enable or disable of the signal.)

Each of these conditions shall be selectively enabled or disabled. In addition, response to an external halt signal, from another processing element, shall selectively enabled or disabled. The address compare shall function on the address field of the processor and shall be capable of indicating a match within two fault tolerant clock cycles of its occurrence.

(b) Selected Exceptions

Each monitor shall be capable of displaying the occurrence of exceptions. A method of signalling an external test device whenever an exception occurs shall be provided. The exceptions implemented shall be documented in the hardware design specification.

(c) External Commands

Each processing element shall have the capability of accepting commands from external test devices. Commands that shall be implemented are: halt on condition; read memory; halt from external condition; start program execution.

(d) The test software shall have the capability to start or stop the real time clock, the interval timer(s), and the asynchronous watchdog timer.

3.5.1.1.3 Common Functions Some elements of the FTP are shared by both the CP and the IOP. These functions either do not have a high usage rate or are used for CP/IOP intercommunication and therefore need not be duplicated. Functions that fall into this category are the data exchange mechanism, shared RAM and shared subroutine memory. All devices of this type reside on a internal shared bus. Both the IOP and CP contend for use of this bus.

3.5.1.1.3.1 FTP Data Exchange Mechanism The data exchange is the means by which data is passed between channels of the FTP. Functions performed are voting and transfers from single channel to all channels. Data is transferred in serial format to minimize the number of signal paths between channels of the exchange. The requirements are:

Word (16 bit) rate: 1 word per 5.25 microseconds

Link Bit Rate: 4 MHz

Maximum Channel Separation (Cable Length): 5m

(1) Internal View

The data exchange appears as six registers. Four of these are write only:

XA	Channel A to all transmitters
XB	Channel B to all transmitters
XC	Channel C to all transmitters
XV	Voter

and two may be read:

XR	Receiver register
XE	Error register.

Data transfers are accomplished by first writing to a transmitter register and then reading the result from the receiver register. For example, if all channels execute a move to XA, channel A's data will appear in XR of all channels. If all channels execute a move to XV, the "bit by bit" "2 out of 3" majority of the three channels' data appear in XR of all channels.

(2) Overview

Figure 17 on page 62 functionally depicts the data exchange mechanisms and paths of all three channels (A,B,C) and the flow from channels (left), through interstages (center) back to channels (right). Within a channel, left to right, are the transmitter register, buffers to the interstage and bidirectional buffers to and from the other channels, the interstage and interstage buffers, crosschannel wiring, vote/selection circuitry, and the receiver and error registers.

Six paths emanate from each channel: two crosschannel, two interstage out, and two voter inputs. The interstages are collo-

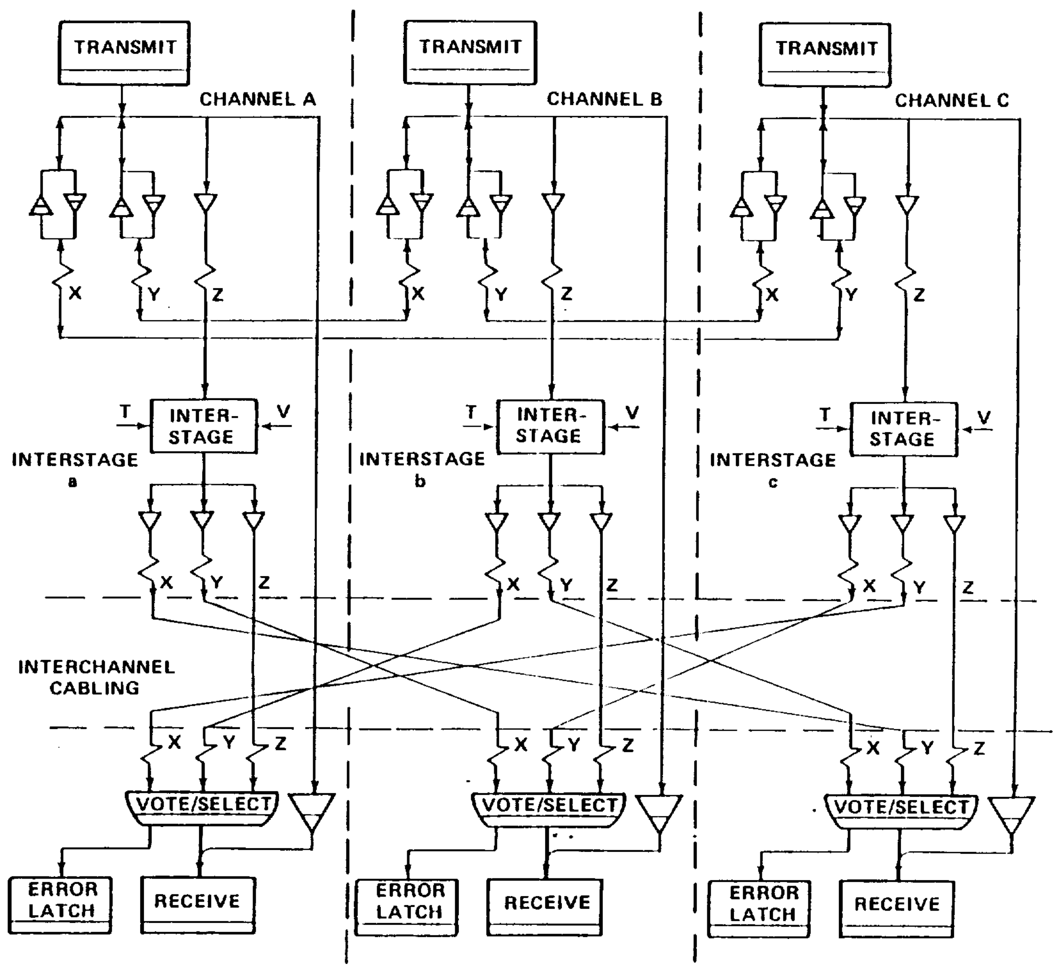

Figure 17. Data Exchange Functional Diagram

cated with their associated channels, so the third interstage output and input paths do not require cabling.

The interstages shall be equipped with independent power (V) and fault tolerant clock reference (T). Resistor-like symbols indicate electrical fault containment.

(3) Congruency

The interstages are provided to insure that two good channels cannot receive differing data from a single source (i.e., the third channel) in the presence of a single fault. If a channel has failed in such a way that it produces ambiguous (incongruent) data, then, by definition, no interstages have failed and all interstages must produce unambiguous (even if incorrect or different) data. Conversely, if an interstage has failed so as to produce ambiguous output, that interstage is the first failure and the three channels and other two interstages are good. In either case, a two of three vote of the interstage outputs produces consistent data in all good receivers.

(4) Exchange

Transfer of data takes place in two phases. During the first phase, data is transmitted to the interstages, and the interstages latch and stabilize the data; during the second phase, the interstage outputs are passed to the channels and each channel performs a two of three vote on the incoming data, storing the result in the receiver register.

(5) Sequencing

A transfer operation is initiated when a write to one of the transmit registers takes place. If the exchange hardware is not busy, it initiates a transaction; if the exchange is busy, the operation is held up by forcing a processor to wait until the exchange is no longer busy.

The following control sequence occurs following a write to the transmit register.

(a) Interstage latches data
(b) Voted interstage latches data
(c) If a single byte operation

Then: Release exchange

Else: Repeat (1) and (2) for second byte

Reading of the receiver register while the exchange is busy will cause the processor to wait until the operation completes.

(6) Error Indications

Disagreements among channel inputs are registered in the error latch XE. The bits indicate which channel is in error and also the operation(s) taking place:

15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
x $\bar{c}$ $\bar{b}$ $\bar{a}$	x $\bar{c}$ $\bar{b}$ $\bar{a}$	x $\bar{c}$ $\bar{b}$ $\bar{a}$	x $\bar{c}$ $\bar{b}$ $\bar{a}$
XV	XC	XB	XA

An error is indicated by a zero in the corresponding bit. The indication remains until XE is read (all bits are set to one by the clear operation). Bits marked X are indeterminate.

(7) Restoration

The receiver register may be written so that an interrupting program may use the exchange and restore XR before returning control.

(8) Synchronization

The data exchange serves as the means for establishing and verifying synchronism between the channels of the FTP. When the exchange is selected as a source while it is busy, the processor will be detained until the operation completes; this aligns the processor instruction stream with the fault tolerant clock. If, further, the received data agrees with its expected value, then the source of the data must be program synchronism. Program synchronization may be established by conducting repeated data exchanges until agreement is found in the received data.

3.5.1.1.3.2 Mass Memory Bus Interface The mass memory bus interface shall provide the means for transmitting and receiving mass memory data. Since the mass memory is on a triply redundant contention bus each of the redundant IOPs must contend for access to the mass memory bus.
Figure 18 on page 65 is a block diagram of the mass memory interface. The following is an explanation of the major elements of the interface.

(1) Voter, Error Latch and Receiver

The receiver shall accept the serial data stream, verify compliance with the network protocol, convert to internal logic levels and timing, and check for transmission errors. If the data does not comply with the protocol, the receiver shall reject the transmission until a valid protocol synchronization is received. If a transmission error is experienced, the event shall be stored in an error register. This error register shall store the errors until the CPU reads and clears it.

In the case of the triplex mass memory bus there is a receiver for each of the three mass memory buses. The outputs from the three receivers are applied to each of the voters (one per redundant

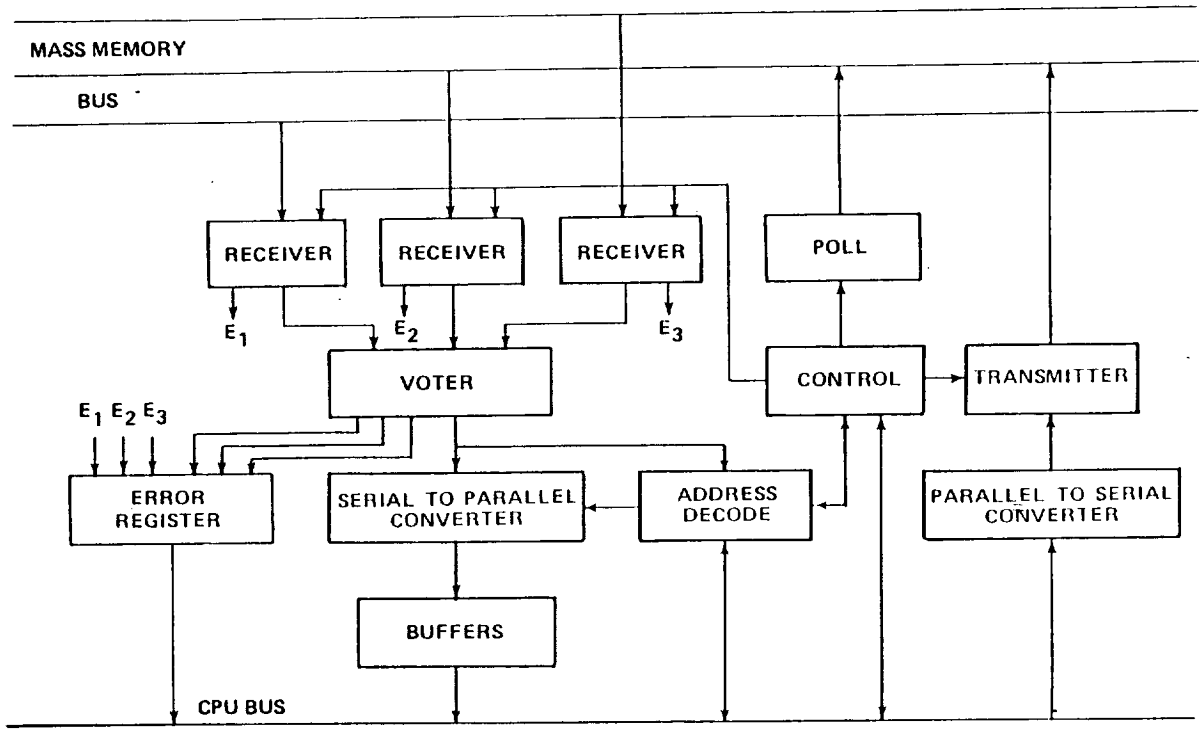

Figure 18. Mass Memory Bus Interface

channel). The voter receives the triply redundant transmission from the mass memory bus and does a bit by bit comparison of the transmission. If any of the bits disagree it shall output the value of the two out of three majority of that bit position. At the same time it shall set a flag in the error latch indicating on which line it perceived an error. The error latch flags are stored within the error register until acted upon by the software.

In the case of a duplex mass memory bus (or triplex downmoded to duplex) the voter will only have two inputs to function with. The output of the voter will always have an error in the nonexistent channel position. This error may be masked by the software. If either of the other channel positions indicates an error, then it will be an indication that there was a disagreement between the two data paths. With only two data copies the voter can detect an error but not isolate it. Isolation of the fault will depend upon the detection of errors at the receiver prior to the voter.

To support receipt of simplex data (including triplex/duplex downmoded to simplex) a bypass of the voter shall be provided. In the simplex configuration, any data source received by the voter may be selected. The bypass shall be constructed so as to insure that a failure causing the bypass mode is detectable.

(2) Data Conversion

The output of the voter is a serial bit stream. This output is applied to the serial to parallel converter. The output of the serial to parallel converter can be read by the CPU through the buffers.

(3) Transmitter

Each channel transmits on one of the redundant mass memory busses. All redundant processors load their mass memory transmit buffer with the data to be transmitted. The output of the buffers are serialized and applied to the transmitter, which if enabled, transmits it.

The transmission length shall be limited by the channel hardware and there shall be a delay in reenabling to protect against babbling mode faults.

(4) Control

The interface shall contain a control module that enables and disables functioning of the interface.

(5) Bus Contention

Each transmission on the mass memory bus is preceded by a contention for the bus among those processing elements that are ready to transmit. The Laning poll technique shall be used for resolving contention. At the end of the previous transmission the protocol receiver and transmitter will be gated off and the contention logic enabled. Any processing element that requires the network will wait a prescribed time after normal transmission ends and start the poll. Any unit not entering the poll within a specified time of the start will be locked out of this poll sequence. Whenever an element observes the completion of the poll, even when it was not a participant, it will switch over to the data mode. This will allow all elements on the network to receive the data transmission, and will be performed totally in hardware so as not to burden the software. The polling is performed simultaneously on all three mass memory busses. When the poll is complete the FTP will do an exchange to verify that all elements have the same poll result.

3.5.1.1.3.3 CP/IOP Communication. The CP and IOP will communicate with one another through a shared memory. Signalling via interrupt requests will provide notification of a waiting message.

(1) Shared Memory

The shared memory is a mechanism for allowing the computational processor and the input output processor to communicate. It is used by both as a mail box to leave messages, requests and data for each other. This memory space shall contain 2048, 16 bit words with an access time of two microseconds or less per word.

(2) Interprocessor Interrupts - The CP and The IOP shall each have an output command that requests an interrupt of the other processor.

3.5.1.1.3.4 Fault Tolerant Clock. Synchronism between channels of the FTP is maintained by the Fault Tolerant Clock (FTC). Signals from FTC elements within each channel are distributed to all channels, each channel's FTC element operating so as to remain in synchronism with a derived majority of the FTC signals. The presence of any single fault within the elements of the FTC will not cause unfailed channels to diverge.

A functional mechanization of the Fault Tolerant Clock is shown in Figure 19 on page 68. Each channel contains an FTC element, consisting of a clock receiver and digital phaselocked loop, which produces internal clocking pulses for that channel and an output signal F. The F's from the three elements are distributed to the interstages, where clocks for the interstages (T) are derived and returned to the channels. Each element then adjusts its own output so as to remain in phase with the majority of the incoming T signals.

It can be shown that a minimum of four clock elements are required to tolerate a single congruency fault in a single level system. In this system, the inclusion of the interstage clock receivers (already needed for other purposes) obviates the need for a fourth clock element.

Fault Tolerant Clock parameters that, for example, may suit the AIPS proof-of-concept system are:

Processor clock frequency:	8 MHz
FTC period (F and T):	2.625 microseconds
Worst case misalignment (any F or T):	0.45 microseconds

In order that Fault Tolerant Clock failures be detectable a means shall be provided for the observation of errors at the element and interstage inputs. It is also required that a means of causing controlled misbehavior be included to demonstrate the availability of the redundant paths and error masking capabilities.

The stability of the individual oscillators shall be 10^{-4} seconds/seconds or better.

3.5.1.1.3.5 Power Conditioning Each channel of the FTP shall contain power conditioning, independent of the other channels. The power conditioning will accept power from the AIPS power system and convert it to suitable voltages to operate the channel.

3.5.1.2 Fault Tolerant Multiprocessor (FTMP)

The fault tolerant multiprocessor is made up of processing elements that can be reconfigured into triplex units. The elements of a triplex unit, or triad, are synchronized to each other. Units can be added, removed or reassigned to triads under control of other triads. In case of failure a

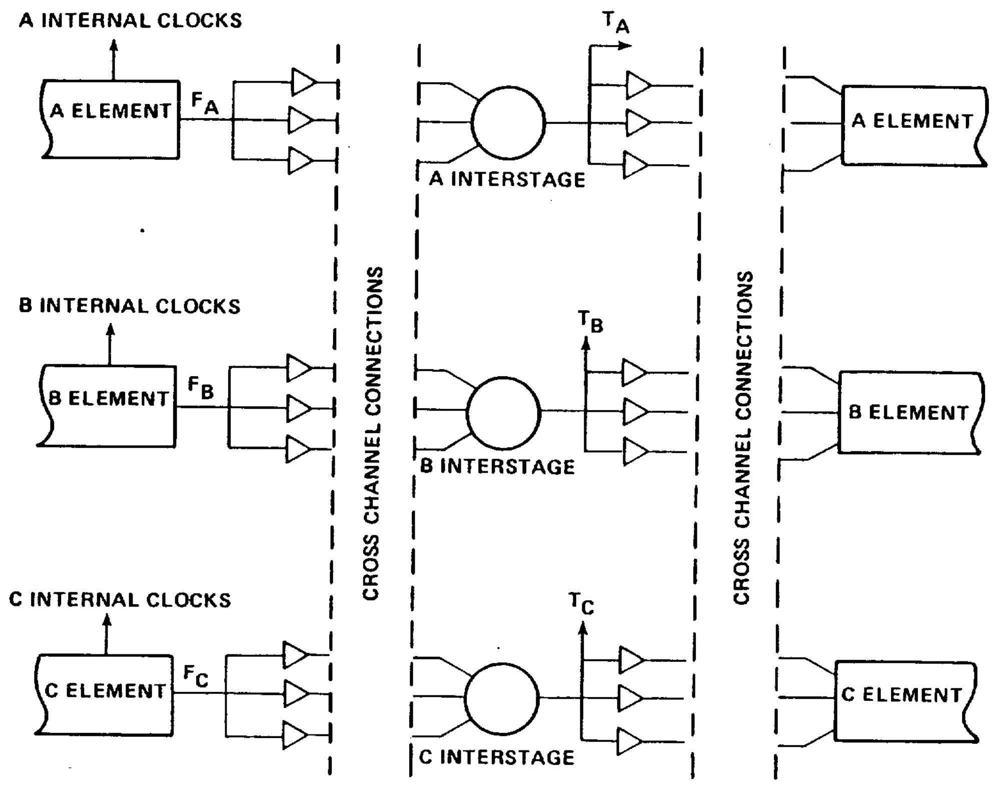

Figure 19. FTP Fault Tolerant Clock Functional Mechanization

unit can be taken completely off line. A typical configuration of an FTMP is shown in Figure 20 on page 69. Each element of a triad is executing identical software in synchronism with the other members of the triad. One member of the FTMP is assigned the task of being the intercomputer (IC) processor triad. Another is assigned the task of being the input/output processor triad. Other members are used as processing elements. These assignments can be changed, under software control, as needed. In the proof of concept design some processors will be capable of being IC controllers, some I/O controllers, some capable of controlling both and others capable of only being computational processors. This is pictured in Figure 21 on page 70. Triads 1 and 4 can be either IC or I/O controllers. Triad 2 can only be an IC controller, while triad 5 can only be an I/O controller. Triad 3 does not have an interface to either, so therefore can only used for computation. The quantity of each for the POC system will be determined by the coverage required. The major elements of an FTMP processor are specified below.

3.5.1.2.1 **Fault Tolerant Clock.** Synchronism between units of the FTMP is supported by the Fault Tolerant Clock (FTC). Signals from FTC elements within each unit are distributed to all units, each unit's FTC element operating so as to remain in synchronism with a derived majority of the

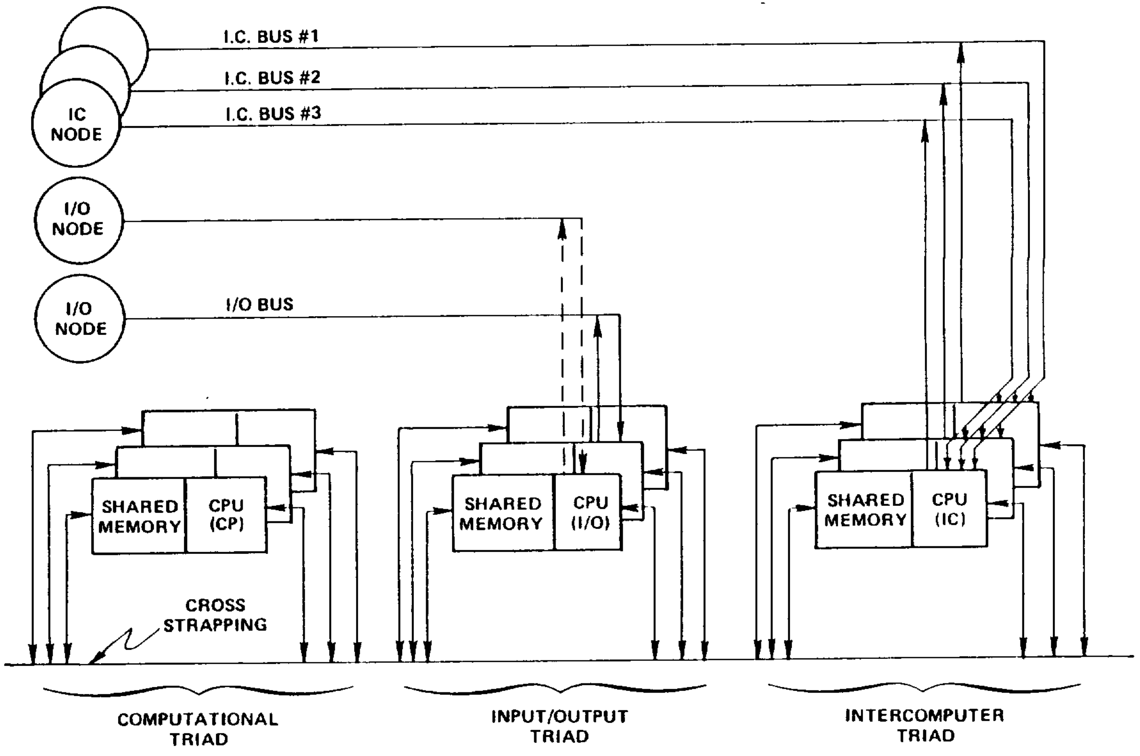

Figure 20. Fault Tolerant Multiprocessor Organization

FTC signals. The presence of any single fault within the elements of the FTC will not cause unfailed channels to diverge.

A functional mechanization of the Fault Tolerant Clock is shown in Figure 22 on page 71. Each unit contains an FTC element, consisting of a clock receiver and digital phase-locked loop, which produces internal clocking pulses for that channel and an output signal F.

It can be shown that a minimum of four clock elements are required to tolerate a single congruency fault in a single level system. In the FTMP, all clock elements have access to the outputs of all others. Four are designated as active by control registers associated with each, and all elements synchronize to the active four. Replacing an active element consists of merely notifying each element which four are to be active.

Fault Tolerant Clock parameters that, for example, may suit the AIPS proof-of-concept system are:

Processor clock frequency:	8 MHz
FTC period (F and T):	2.625 microseconds

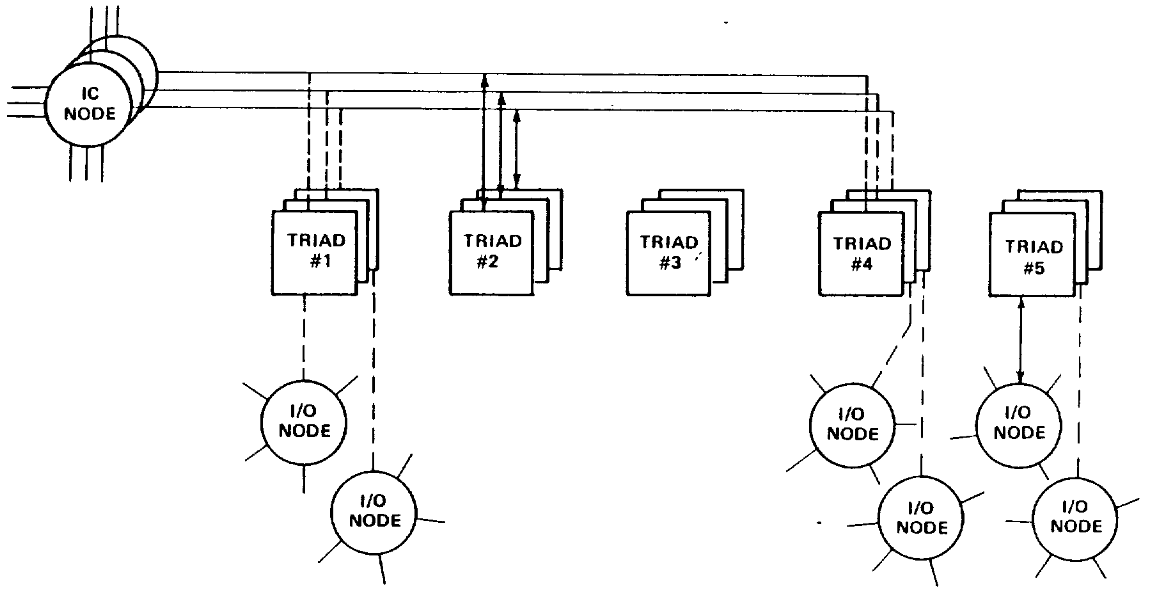

Figure 21. Fault Tolerant Multiprocessor Assignments

Worst-case misalignment (any F or T):	0.45 microseconds

In order that Fault Tolerant Clock failures be detectable a means shall be provided for the observation of errors at the element and interstage inputs. It is also required that a means of causing controlled misbehavior be included to demonstrate the availability of the redundant paths and error masking capabilities.

The stability of the individual oscillators shall be 10^{-4} seconds/seconds or better.

3.5.1.2.2 FTMP Processing Element

Each processing element shall be comprised of the elements as shown in Figure 23 on page 72.

3.5.1.2.2.1 Central Processor Unit The central processor unit (CPU) is the computational processing element. All data processing and manipulation is performed by the CPU. It will operate with a clock speed appropriate for the chosen processor and the processor shall function with virtual memory through a memory management unit. Memory accesses that are not resident within the local storage area will cause page faults. The processor shall be capable of suspending processing, reloading memory with the required data and restarting program execution without loss of continuity. Upon power up, all CPUs will be reset. This reset shall clear

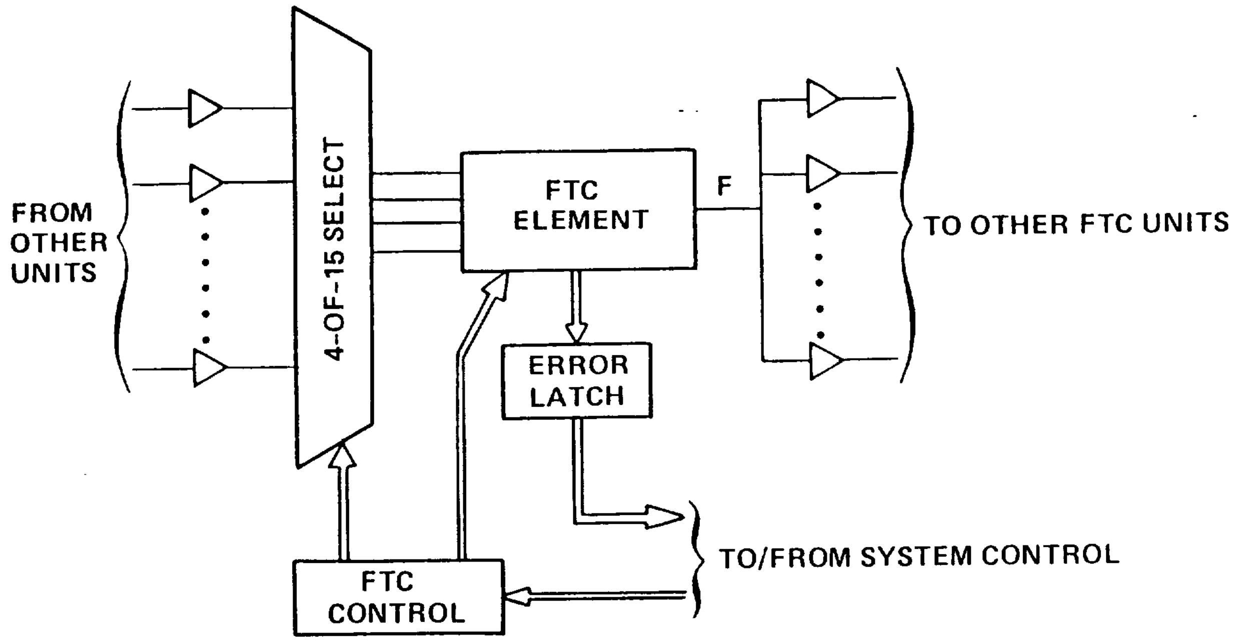

Figure 22. FTMP Fault Tolerant Clock Functional Mechanization

all working registers and flags. As part of the initialization the CPU shall reset all interfaces to a predetermined state.

3.5.1.2.2.2 Memory The memory module contains the local storage for a processing module. It contains all program storage, constants and variables necessary to perform the assigned tasks. Each memory shall be comprised of random access memory (RAM) and read only memory (ROM). The total of RAM and ROM is 128k bytes.

(1) RAM

The RAM may be used for the storage of task programs assigned to the triad of which this processor is a member. Programs are loaded into RAM from shared memory. The RAM shall have an access time such that no more than two processor wait cycles are required for access.

(2) ROM

The ROM shall contain the fixed operational programs, subroutines, constants, initialization and diagnostic programs. The ROM should have an access time such that no more than two processor wait cycles are required for access.

3.5.1.2.2.3 Timers The timers are used for keeping track of real time, timing tasks, and maintaining a check on the health of the hardware and software. Each timer module shall contain a real time clock, an interval timer(s), and a watchdog timer.

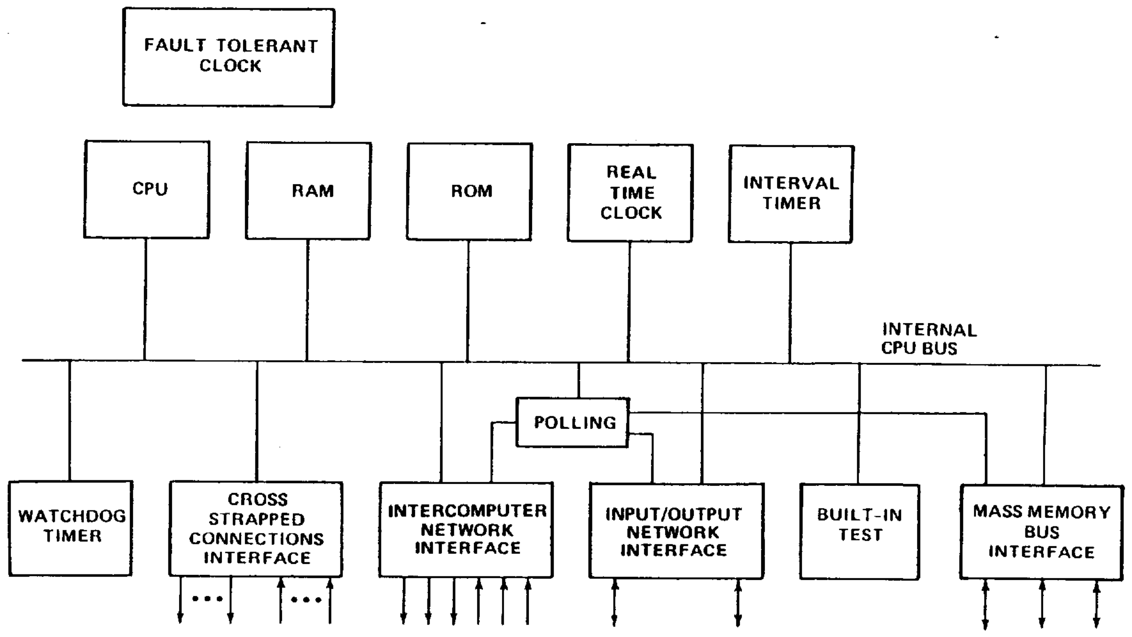
FAULT TOLERANT CLOCK
CPU
RAM
ROM
REAL TIME CLOCK
INTERVAL TIMER
INTERNAL CPU BUS
POLLING
WATCHDOG TIMER
CROSS STRAPPED CONNECTIONS INTERFACE
INTERCOMPUTER NETWORK INTERFACE
INPUT/OUTPUT NETWORK INTERFACE
BUILT-IN TEST
MASS MEMORY BUS INTERFACE

Figure 23. FTMP Typical Processing Module

(1) Real-Time Clock

The real time clock shall be a 32 bit hardware register which can be preset to the system time under program control. Once set the real time clock will increment every 50 to 100 microseconds (increment to be specified during preliminary design). When the count reaches the maximum value it will reset to zero and continue to count up. Each memory element within the FTMP shall have a real-time clock. All processor triads shall have access to the real-time clock via the multiprocessor cross strap bus. When a triad requires the real time it will be available by reading the memory location assigned to the clock.

(2) Program Interval Timer(s)

The interval timer(s) shall be a 16 bit hardware register that can be preset to any value by the software. Once set the interval timer shall decrement to zero and stop. At this time an interrupt is requested. The interval timer(s) shall be capable of being read or written. The interval timer(s) shall have a resolution between 8 and 24 microseconds based upon available timing.

(3) Asynchronous Watchdog Timer

The asynchronous watchdog timer is a fault detection mechanism which provides an overall check on the health of the processing element. Under normal operating conditions the software must reset it periodically. This verifies that the system is functional. If not reset or a hardware failure prohibits service, the asynchronous watchdog timer shall reset the CPU. It shall be possible to inhibit the asynchronous watchdog timer when operating in the test mode. The asynchronous watchdog timer shall be independent of the FTMP oscillators.

3.5.1.2.2.4 **Built-in Test Features** The system component shall have built-in test features to provide visibility into the system operation. The following features shall be included.

(1) Dual Port Memory

A dual port memory shall provide a communication path between the Proof-of-Concept system and the Integration and Evaluation Facility host computer. The facility host computer shall be capable of sequentially reading the data stored in all dual port memories in all elements of the system. Additionally, it shall be possible for the facility host computer to download programs and commands to the processing elements without use of the communication networks. This dual port memory will appear to the processing elements and the facility host computer as 2048 words of RAM memory.

(2) Bus Monitor

Each processing element shall have a nonintrusive bus monitoring capability. This monitoring capability should be capable of detecting specified events within the processor. This capability shall reside on an independent circuit board such that the feature may be installed or deleted as required. Detectable events shall be:

(a) Condition Compare

Each monitor shall have an address and data compare register which can be read or written by either the processor or through the external test device, such as the the facility host computer. The address compare register will be used to continually monitor the address bus. Whenever an address match occurs, several results shall be possible.

(i) A flag will be set indicating a match; This flag will be accessible by the facility host computer through the dual port memory.

(ii) An interrupt will be provided and if enabled a "halt" of the processing element will occur.

(iii) A global halt signal shall be transmitted to other processing elements. (The results at other sites will depend upon the enable or disable of the signal.)

Each of these conditions shall be selectively enabled or disabled. In addition, response to an external halt signal, from another processing element, shall be selectively enabled or disabled. The address compare shall function on the address field of the processor and shall be capable of indicating a match within two fault tolerant clock cycles of its occurrence.

(b) Selected Exceptions

Each monitor shall be capable of displaying the occurrence of exceptions. A method of signalling an external test device whenever an exception occurs shall be provided. The exceptions implemented shall be documented in the hardware design specification.

(c) External Commands

Each processing element shall have the capability of accepting commands from external test devices. Commands that shall be implemented are: halt on condition; read memory; halt from external condition; start program execution.

(d) The test software shall have the capability to start or stop the real time clock, the interval timer(s), and the asynchronous watchdog timer.

3.5.1.2.2.5 Fault Tolerant Cross Strapped Connections Interface

Each FTMP processor shall be connected to all shared memory elements by dedicated communication paths. That is, a processor will have a private connection to all shared memory elements. Whenever a processing element transmits, it transmits to all shared memory elements simultaneously. It is necessary, when an element (processor or shared memory element) is configured in a triad, to know which of its communication receivers are attached to other triad elements. The organization of the connection interface is shown in Figure 24 on page 75. The following is a description of how this interface shall function.

(1) Configuration and Control Registers

All members shall have configuration registers to identify triad membership. This configuration shall be nonvolatile. Upon power-up, one of the triads must run a system configuration task. Triad membership is transmitted to all elements by address. Part of this triad membership contains the bus assignments to be used for voting. The configuration register then configures the three of N receiver to select only the buses needed for voting.

(2) Three-of-N Receiver

The three-of-N receiver: receives data, converts logic levels, and supplies the three busses specified by the configuration register to the voter.

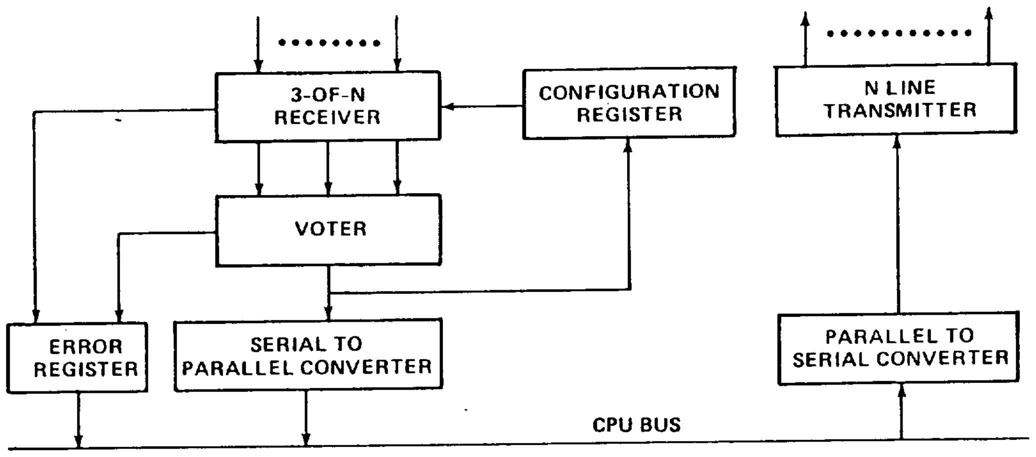

Figure 24. Cross Strapped Connection Interface

(3) Voter and Error Latch

The voter takes the three serial buses from the receiver and does a bit by bit comparison of the transmission. If any of the bits disagree it will only output the majority of that bit position. At the same time it will set a flag in the error latch indication on which line it perceived an error. This flag is stored within the error latch until acted upon by the software.

(4) Data Conversion

All elements must contain the necessary logic to convert the voted data from a serial data stream to a parallel word for the CPU to read. This is performed in the serial to parallel converter. Additionally, when the CPU has data to be sent to other members, memories or I/O it must create a serial data stream from the CPUs parallel data word. This is performed in the parallel to serial converter.

(5) Transmitter

The transmitter takes the output of the parallel to serial converter encodes it into the signalling protocol, converts it to the required voltage level and transmits it onto the output lines.

3.5.1.2.2.6 Intercomputer Network Interface

Some of the FTMP processing elements, the number to be determined during preliminary design, shall contain interfaces to the intercomputer network. Those that contain the interface can be configured in triads that can control the IC network. This is shown in Figure 21 on page 70 as a triad interface to the IC Node. A detail of the major components of this interface is shown in Figure 25 on page 76. Each IOP listens to all three

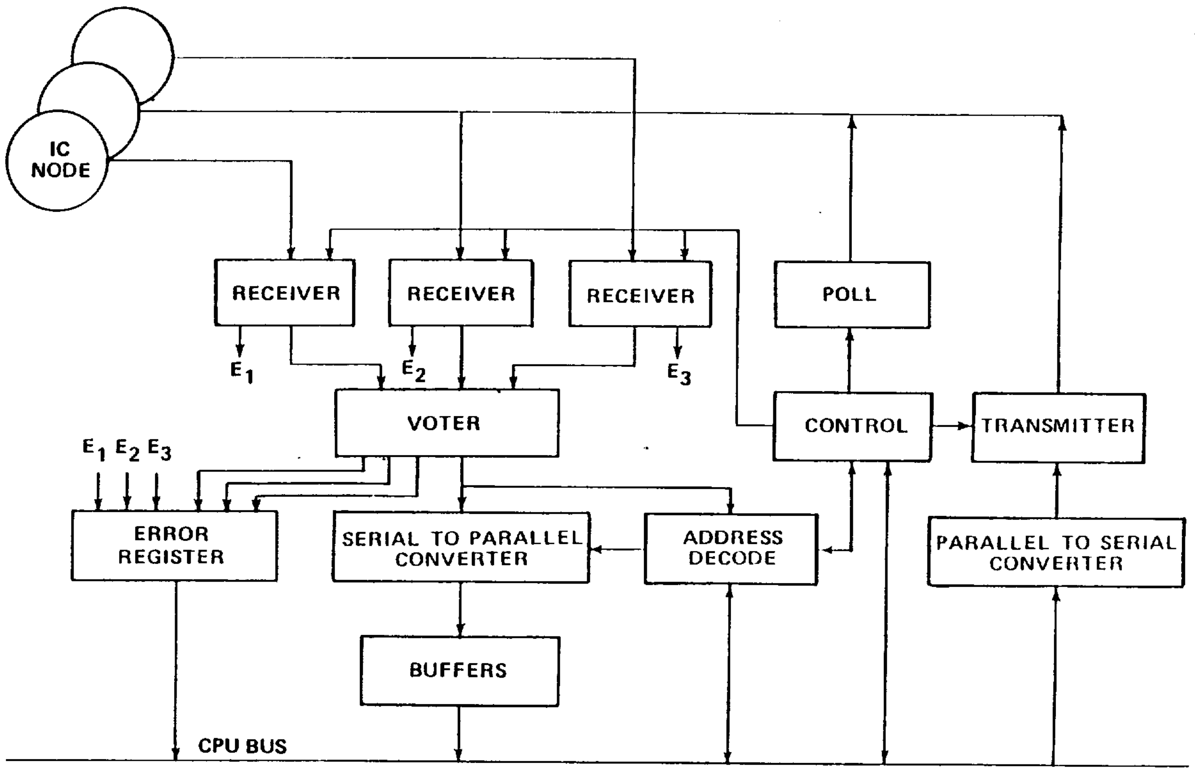

Figure 25. FTMP Intercomputer Network Interface

IC buses, but transmits on one. The following describes the function of each interface element.

(1) Voter, Error Latch and Receiver

The receiver shall accept the serial data stream, verify compliance with the network protocol, convert to internal logic levels and timing, and check for transmission errors. If the data does not comply with the protocol, the receiver shall reject the transmission until a valid protocol synchronization is received. If a transmission error is experienced, the event shall be stored in an error register. This error register shall store the errors until the CPU software reads and clears it.

In the case of the triplex intercomputer network there is a receiver for each of the three IC buses. The outputs from the three receivers are applied to each of the FTMP IOP voters (one per redundant channel). The voter receives the triply redundant transmission from the Intercomputer network and does a bit by bit comparison of the transmission. If any of the bits disagree it shall output the value of the two out of three majority of that bit position. At the same time it shall set a flag in the error latch indicating on which line it perceived an error. The error latch flags are stored within the error register until acted upon by the software.

In the case of a duplex intercomputer network (or triplex downmoded to duplex) the FTMP voter will only have two inputs. The output of the voter error latch will always indicate an error in the nonexistent channel position. If either of the other channel positions indicates an error, then it will be an indication that there was a disagreement between the two data paths. With only two data copies the voter can detect an error but not isolate it. Isolation of the fault will depend upon the detection of errors at the receiver prior to the voter.

In support of simplex (or triplex/duplex downmoded to simplex) a bypass of the voter shall be allowed. In the simplex configuration any data source received by the voter may be selected, by the software, for use instead of the voter output. The bypass shall be constructed so as to insure that a failure causing the bypass mode is detectable. Congruency is performed by doing a simplex write to the FTMP shared memory.

(2) Data Conversion

The output of the voter is a serial bit stream that has had vote discrepancies masked. This output is then applied to a serial to parallel converter. The output of the serial to parallel converter can be read by the CPU through the buffers.

(3) Address Decode

Each subscriber that is connected to the intercomputer bus shall have a unique address for identification. All functional subscribers shall receive all messages that appear on the intercomputer network. It is the responsibility of the subscriber to determine which messages are intended for it. This is the function of the address decoder. As each message is received, the address decoder will look at the address portion of the message and if it finds a match notify the CPU of a waiting message. It shall be possible for the address decoder to respond to broadcast messages as well as specific messages.

(4) Transmitter

Each channel of an FTMP is enabled to transmit on one of the intercomputer buses. Each CPU loads the parallel to serial converter with a message to be sent. This serial bit stream is then applied to the transmitter, which encodes the message into the network protocol and transmits on the assigned bus.

The transmission length shall be limited by the channel hardware and there shall be a delay in reenabling to protect against babbling mode faults.

(5) Control

The interface shall contain a control module that enables and disables functioning of the interface. It also provides the sequence control for the protocol encoder and decoder and is under software control of the CPU.

(6) Network Contention

Each transmission on the intercomputer network is preceded by a contention for the network among those processing elements that are ready to transmit. The Laning poll technique shall be used for resolving contention. At the end of the previous transmission the protocol receiver and transmitter will be gated off and the contention logic enabled. Any processing element that requires the network will wait a prescribed time after normal transmission ends and start the poll. Any unit not entering the poll within a specified time of the start will be locked out of this poll sequence. Whenever an element observes the completion of the poll, even when it was not a participant, it will switch over to the data mode. This will allow all elements on the network to receive the data transmission, and will be performed totally in hardware so as not to burden the software. The polling is performed simultaneously on all three IC buses. When the poll is complete the FTMP will do an exchange using the shared memory to verify that all elements have the same poll result.

3.5.1.2.2.7 Input/Output Network Interface

Some of the FTMP processing elements, as determined during preliminary design, shall contain an interface to the input output network. Those elements that contain the interface will be configured into triads that can be assigned the task of controlling the I/O network. This is shown in Figure 26 on page 79. The I/O network is simplex and therefore can only have one transmitter enabled at a time. However, all members of the triad shall listen and compare results to assure source congruency. The module that is selected to transmit can be disabled and another enabled by the other triads in case of a failure. The major elements of the interface are described below.

(1) Receiver

The receiver shall accept the serial data stream, verify compliance with the network protocol, convert to internal logic levels and timing, and check for transmission errors. There is a receiver for the I/O bus in each triad member.

(2) Data Conversion

The output of the receiver is a serial bit stream. This output is applied to a serial to parallel converter. The output of the serial to parallel converter can be read by the CPU through the buffers.

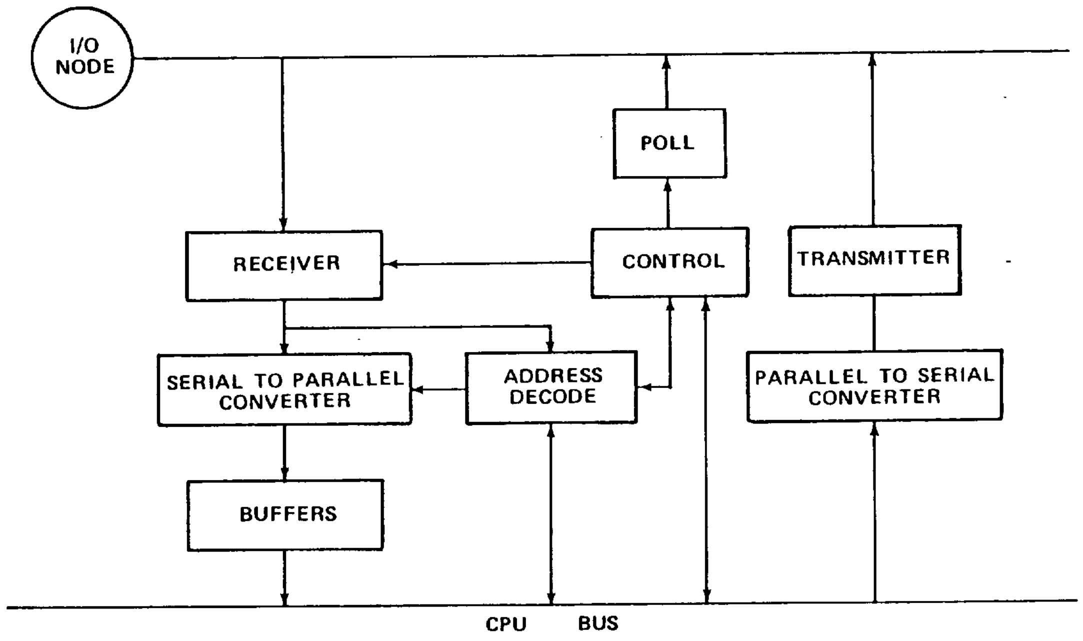

Figure 26. FTMP Input/Output Bus Interface

(3) Address Decode

Each device that is connected to the input output bus shall have a unique address and a broadcast address for identification. All functional devices receive all messages that appear on the input output network. It is the responsibility of the device to determine which messages are intended for it. This is the function of the address decoder. As each message is received, the address decoder must look at the address portion of the message and if it finds a match notify the CPU of a waiting message. It should be possible for the address decoder to respond to broadcast messages as well as specific messages.

(4) Transmitter

Only one member of the triad is enabled to transmit on the input output bus. However, since all members of the triad are executing the same software, each CPU loads the parallel to serial converter with a message to be sent. This serial bit stream is then applied to the transmitter encoder, which encodes the message into the network protocol. The output of the encoder is applied to the input of the transmitter. Only one transmitter is enabled, but this is invisible to the CPU.

The transmission length shall be limited by the channel hardware and there shall be a delay in reenabling to protect agains babbling mode faults.

(5) Control

The interface should contain a control module that enables and disables functioning of the interface. It also provides the sequence control for the protocol encoder and decoder and is under software control of the CPU.

(6) Network Contention

Each transmission on the input/output network is preceded by a contention for the network among those processing elements that are ready to transmit. The Laning poll technique shall be used for resolving contention. At the end of the previous transmission the protocol receiver and transmitter will be gated off and the contention logic enabled. Any processing element that requires the network will wait a prescribed time after normal transmission ends and start the poll. Any unit not entering the poll within a specified time of the start will be locked out of this poll sequence. Whenever an element observes the completion of the poll, even when it was not a participant, it will switch over to the data mode. This will allow all elements on the network to receive the data transmission, and will be performed totally in hardware so as not to burden the software.

3.5.1.2.2.8 Mass Memory Bus Interface The mass memory bus interface shall provide the means for transmitting and receiving mass memory data. Since the mass memory is on a triply redundant contention bus each processor must contend for access to the mass memory bus.
Figure 27 on page 81 is a block diagram of the mass memory interface. The following is an explanation of the major elements of the interface.

(1) Voter, Error Latch and Receiver

The receiver shall accept the serial data stream, verify compliance with the network protocol, convert to internal logic levels and timing, and check for transmission errors. If the data does not comply with the protocol, the receiver shall reject the transmission until a valid protocol synchronization is received. If a transmission error is experienced, the event shall be stored in an error register. This error register shall store the errors until the CPU reads and clears it.

In the case of the triplex mass memory bus there is a receiver for each of the three mass memory buses. The outputs from the three receivers are applied to each of the voters (one per redundant channel). The voter receives the triply redundant transmission from the mass memory bus and does a bit by bit comparison of the transmission. If any of the bits disagree it shall output the value of the two out of three majority of that bit position. At the same time it shall set a flag in the error latch indicating on which line it perceived an error. The error latch flags are stored within the error register until acted upon by the software.

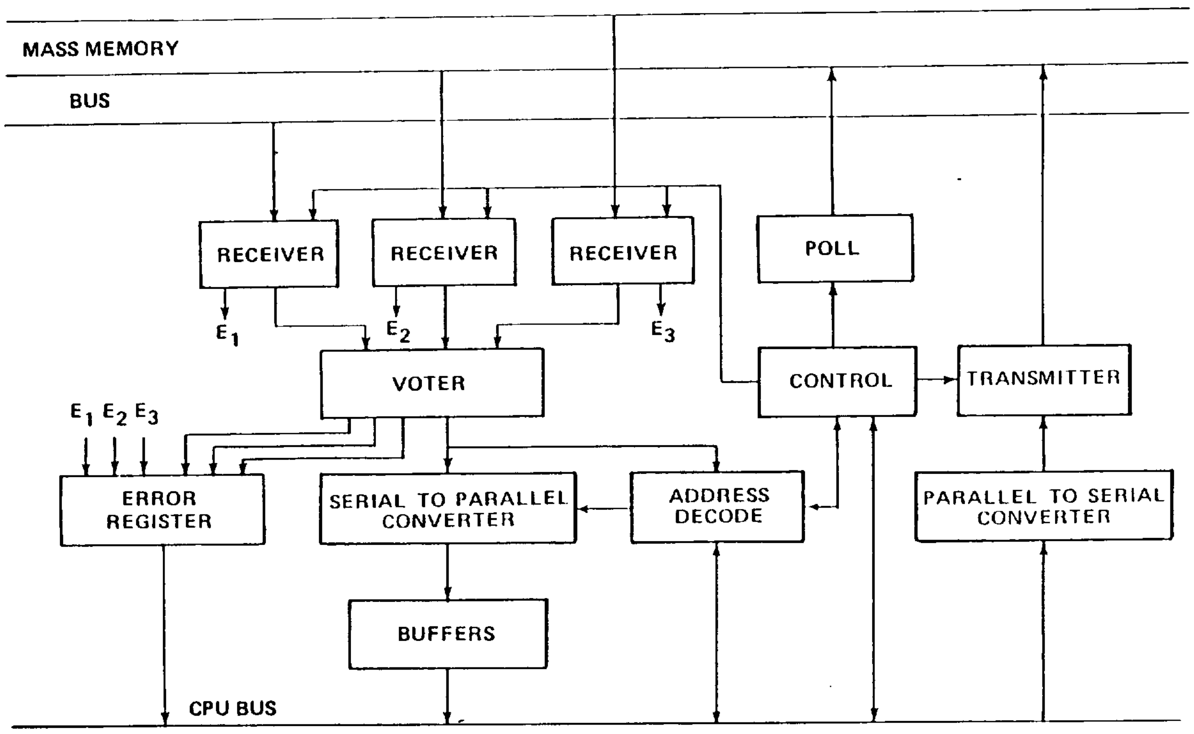

Figure 27. Mass Memory Bus Interface

In the case of a duplex mass memory bus (or triplex downmoded to duplex) the voter will only have two inputs to function with. The output of the voter will always have an error in the nonexistent channel position. This error may be masked by the software. If either of the other channel positions indicates an error, then it will be an indication that there was a disagreement between the two data paths. With only two data copies the voter can detect an error but not isolate it. Isolation of the fault will depend upon the detection of errors at the receiver prior to the voter.

To support receipt of simplex data (including triplex/duplex downmoded to simplex) a bypass of the voter shall be allowed. In the simplex configuration, any data source received by the voter may be selected. The bypass shall be constructed so as to insure that a failure causing the bypass mode is detectable. Congruency is performed by doing a simplex write to the FTMP shared memo memory.

(2) Data Conversion

The output of the voter is a serial bit stream that has had vote discrepancies masked. This output is then applied to a serial to parallel converter. The output of the serial to parallel converter can be read by the CPU through the buffers.

(3) Transmitter

Each channel transmits on one mass memory bus level. All redundant CPUs load their mass memory transmit buffer with the data to be transmitted. The output of the buffers are serialized and applied to the transmitter and if enabled transmits it.

The transmission length shall be limited by the channel hardware and there shall be a delay in reenabling to protect against babbling mode faults.

(4) Control

The interface shall contain a control module that enables and disables functioning of the interface.

(5) Bus Contention

Each transmission on the mass memory bus is preceded by a contention for the bus among those processing elements that are ready to transmit. The Laning poll technique shall be used for resolving contention. At the end of the previous transmission the protocol receiver and transmitter will be gated off and the contention logic enabled. Any processing element that requires the network will wait a prescribed time after normal transmission ends and start the poll. Any unit not entering the poll within a specified time of the start will be locked out of this poll sequence. Whenever an element observes the completion of the poll, even when it was not a participant, it will switch over to the data mode. This will allow all elements on the network to receive the data transmission, and will be performed totally in hardware so as not to burden the software. The polling is performed simultaneously on all three mass memory busses. When a poll is completed the triad will do an exchange to verify that all membersof the triad have the same poll result.

3.5.1.2.2.9 **Registers** Each processing element contains registers that can be accessed from the cross strapped interconnections and appear as locations on the shared memory space. These registers are written into by triads and can be read by the processor on its internal CPU bus.

(1) Interprocessor Communication Registers (IPC) - The interprocessor communication registers are used by triads to send messages to other triads or CPUs directly.

(2) Triad ID - The triad id register is written by the triad that is performing the configuration task. This register will contain the triad membership that this processor has been assigned to.

(3) Bus Assignment - The bus assignment register is written by the triad that is performing the configuration task. This register will

contain the identification of the buses that the triad is to use for voting. Data that is received on these buses will be assumed to be triad specific and be used for voting.

3.5.1.2.3 Shared Memory

Each FTMP contains a shared memory that is used for program and data storage.

In addition to the memory array, the shared memory contains control registers that are addresses within the memory space. These control registers can be accessed from the cross strapped buses and individually from the processor or memory internal bus to which it is connected. Within the shared memory these registers are written by processor triads and are used to identify which memory triad a memory is assigned to. In addition the memory is informed of the current makeup of the processor triads for receiver voting and transmitter selection, and memory relocation for address space definition. Within the memory space there are a set of registers that the processors use. They include a triad identification register for membership inclusion and interprocessor communication registers which allow processors to communicate directly without using the shared memory as an intermediary. Shared memory consists of a number of elements. The memory elements are assembled into triads by the processor triads. Each memory triad is assigned to a memory space. The shared memory shall be designed such that any processing triad appears to have dedicated use of the memory. The organization of the shared memory is shown in Figure 28 on page 83. The shared memory controller shall provide "test and set" capability for any word or bit in read/write memory. In addition, the shared memory controller shall provide a capability for processor triad to processor triad communication, including "notification that a processor to processor communication has occurred." The major elements of the shared memory are described below.

(1) Memory Devices

Each shared memory contains memory devices which can be RAM or ROM or a combination of both. Each memory element will contain 256k bytes.

(2) Receivers

Each memory element contains a receiver which receives the transmission from each processor. The receiver provides any necessary logic level conversion and drive capabilities. The output is applied to the three - of - N selector.

(3) Three-Out-of-N Selector

Each memory shall contain the same number of three-out-of-N selectors as there are triads. This is necessary so that each triad can be identified and function as if the shared memory were dedicated to it. The memory receives the processor triad configuration commands and dedicates a selector to only look at the three lines coming from that processing triad. Those three lines only are then

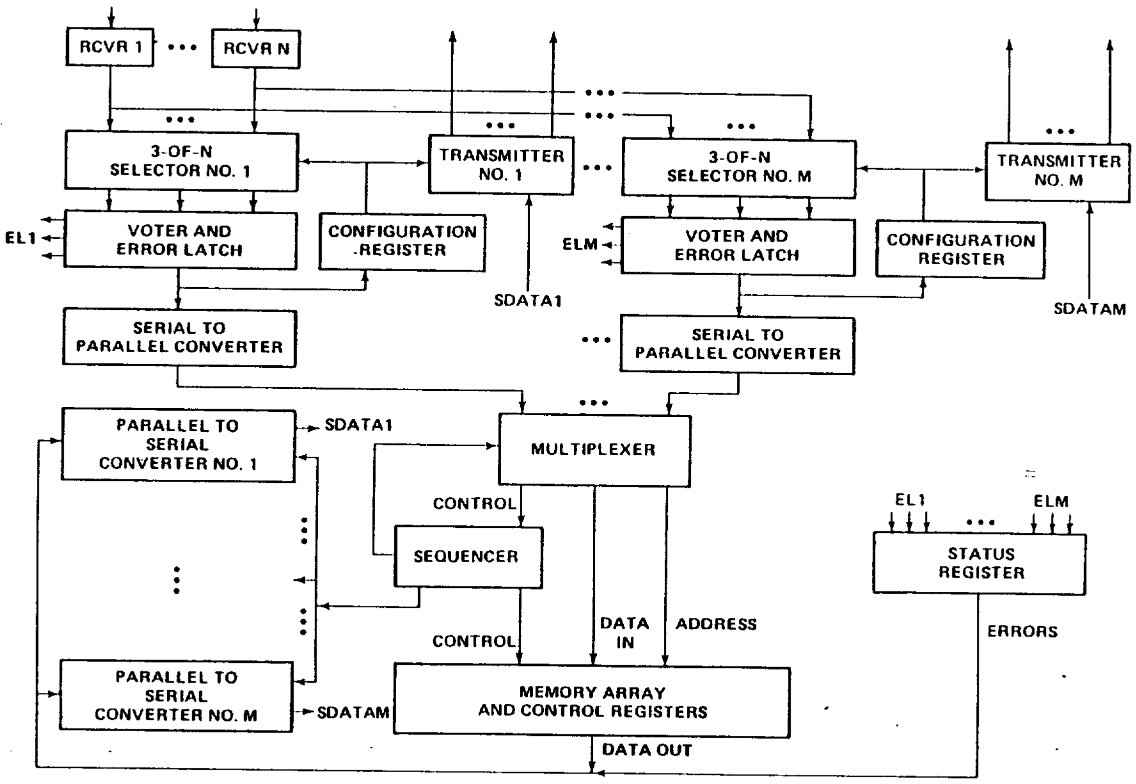

Figure 28. FTMP Shared Memory

passed through to the voter connected to this selector. Therefore the voter will only see data from one triad.

(4) Voter and Error Latch

The outputs of the three-out-of-N selector are first deskewed to correct for transmissions delay differences, and then applied to the input of the voter. The voter performs a bit by bit comparison on the data. If any bit differs, the output of the voter will be the majority of the two out of three in agreement. In addition when a difference is detected a record is kept upon which line this error occurred. The output of the voter is then applied to a serial to parallel converter, and the results of the error latch "ORed" into a status register.

(5) Serial to Parallel Converter

The voted output is entered into a serial to parallel converter to assemble the word into parallel form. When a full message has been received the converter sets a flag for the sequencer to request service. The serial to parallel converter must also hold the data while waiting for the actual access to the memory. The output of the converter is applied to a data multiplexer.

(6) Multiplexer

The multiplexer, under control of the sequencer, selectively applies the address and data (if a write) to the memory devices. At the same time the control portion of the message is decoded and used to determine the reason for the access. The multiplexer must present the data for the proper time to perform the function.

(7) Parallel to Serial Converter

There is a parallel to serial converter provided for every processor triad. The parallel to serial converter is used when ever a processor triad requests a read operation. The data is fetched from the memory devices and loaded into the appropriate serial to parallel converter for transmitting to the requesting triad. Once the data is loaded into the converter the memory is free to perform the next request.

(8) Transmitter

Each transmitter section is assigned to a processing triad through the configuration register which only enables the lines to its processor triad. This is necessary so that the memory can communicate with multiple processors simultaneously. The data is converted to the line protocol and signaling levels and transmitted on the three appropriate lines.

(9) Sequencer

The sequencer is the main controller for the memory. It shall accept the triad commands for the memory. Assign a three out of N receiver and transmitter pair for each processor triad as they are assigned. It receives the message in flags from the serial to parallel converters and schedules service for that triad. It must also examine the control portion of the message to schedule the required task.

(10) Status Register

The status register contains the error latch information for all the voters. It can be read and cleared by the any triad.

(11) Memory Relocation Register

The relocation register is written by a triad to indicate the addresses this memory module is to respond to. It will then map the address into the physical address within the module.

(12) Bus Assignment Registers

The bus assignment registers contain the current processing triad membership and the buses that they are to use. The memory uses the bus assignment register to assign its input receiver/transmitter and data selection circuits to the corresponding triads.

3.5.1.3 Intercomputer Network Node and Interconnections

The intercomputer network is comprised of nodes arranged in simplex, duplex, or triplex layers. The nodes within a layer are connected by links. There is no interconnection between layers at a node. A node is a communication switching point with five input output ports. Figure 29 on page 86 is a basic representation of a node. The internal construction of each port of a node is shown in Figure 30 on page 87. Since a node does not have knowledge of the configuration of the network its receivers must always be enabled. Transmitters are enabled or disabled upon command. As a message is received, it will be regenerated and retransmitted on all enabled ports. At the same time, the message is decoded within the node. If the message is addressed to the node it will respond to the command embedded within the data. If the message is addressed elsewhere it will check for a valid transmission, latch observed error conditions and reset the receiver for the next transmission. When status is requested from the node, the observed errors will be transmitted within the status message.

Hardware will be used to inhibit selected ports from responding to or generating reconfiguration commands. The intent is to inhibit certain designated, attached processors from being able to reconfigure any node. Reconfiguration commands generated by those processors will be ignored by the attached node and not transmitted to any other node. Ports (whether active or not) that are not inhibited will be capable of receiving and processing reconfiguration commands. Reconfiguration commands enable or disable selected port transmitters. Ports will be reconfigured whenever a legal command is received.

Some components are unique to a port (transmit/receive) and some are shared by all the ports (control). The following is a description of the basic components of a node.

3.5.1.3.1 Transmit/Receive Components

3.5.1.3.1.1 Receiver The receiver accepts the signal level on a link and converts it to the internal logic level of the port. The receiver also isolates the node from electrical failures of the link.

3.5.1.3.1.2 Protocol Decoder The protocol decoder accepts the serial data stream from the receiver, checks for protocol compliance and transmission induced errors. The errors are stored for transmission to GPCs when requested. It then synchronizes with the external transmission clock and assembles the message into parallel words. These parallel words are temporarily stored for the control section to examine.

3.5.1.3.1.3 Address Decoder The address decoder compares the address portion of the message with its own internally stored address. If the address matches the decoder signals the control section that there is a

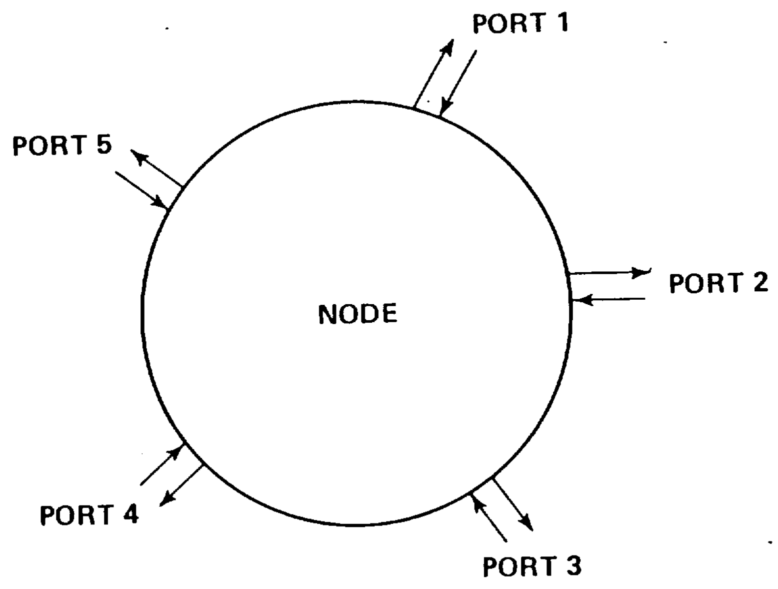

Figure 29. Intercomputer Network Node

message waiting to be processed. If the address does not match the decoder section is cleared to wait for the next message.

3.5.1.3.1.4 **Data Decoder** If a message is addressed to the node the data decoder is used to extract the type of command from the data part of the message. The decoded command is sent to the control section.

3.5.1.3.1.5 **Signal Regeneration Logic** The signal regeneration logic is used to reconstruct the fidelity of the transmission. This is necessary because of the modification to a signal as it passes through circuit elements. After several transitions through circuit elements the transmission could appear to be modified. The input to the regeneration logic is the OR of all the ports. The output of the regeneration logic is applied to the input of the port transmitter.

3.5.1.3.1.6 **Transmitter** The transmitter converts the output of the protocol encoder or regeneration logic into the signaling levels used on the links. It can be enabled or disabled by the port configuration register.

3.5.1.3.2 **Control Components**

3.5.1.3.2.1 **Node Sequencer and Control** The node sequencer and control provides the central control for node operations. It accepts inputs from the address decoder indicating a message for the node has been received. It then decodes the command. There are two types of commands received by the node; 1)Reconfiguration commands and 2) Status requests. Configuration commands are passed to the port configuration register to control the ports. Status and error messages when requested by the processing site

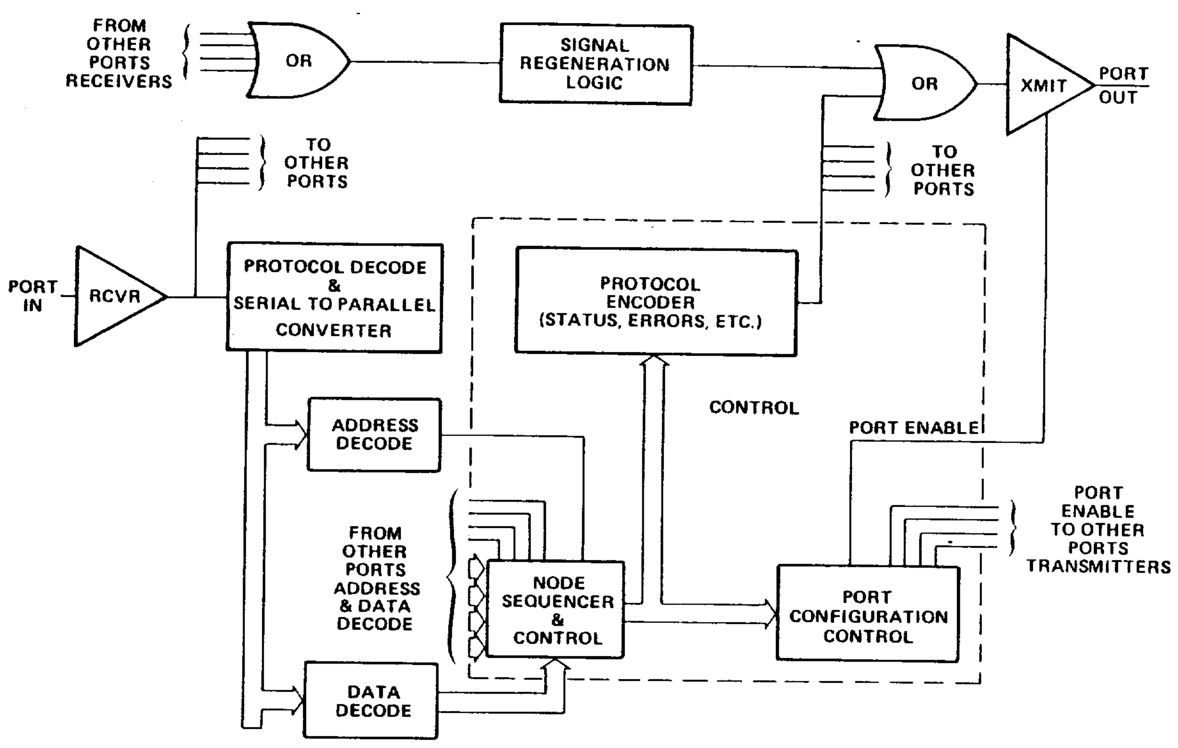
FROM OTHER PORTS RECEIVERS
OR
SIGNAL REGENERATION LOGIC
OR
XMIT
PORT OUT
TO OTHER PORTS
TO OTHER PORTS
PORT IN
RCVR
PROTOCOL DECODE & SERIAL TO PARALLEL CONVERTER
PROTOCOL ENCODER (STATUS, ERRORS, ETC.)
CONTROL
PORT ENABLE
ADDRESS DECODE
FROM OTHER PORTS ADDRESS & DATA DECODE
NODE SEQUENCER & CONTROL
PORT CONFIGURATION CONTROL
PORT ENABLE TO OTHER PORTS TRANSMITTERS
DATA DECODE

Figure 30. Intercomputer Network Node Port

GPCs are passed to the protocol encoder. All control functions necessary to operate the encoder and decoder and the generation of timing signals for the node are generated within the sequencer.

3.5.1.3.2.2 Port Configuration Control The port configuration control accepts the decoded reconfiguration commands from the sequencer and enables or disables the individual port transmitters as commanded. The last command is stored until rewritten by the next command. This register can be read into the encoder if the status of the node is requested.

3.5.1.3.2.3 Protocol Encoder The protocol encoder receives the node response to status requests and encodes it into the link protocol. The output of the encoder is sent to the transmitters.

3.5.1.4 Input Output Network Node and Interconnections

The input output network is comprised of simplex nodes. Nodes are interconnected by links. A node is a communication switching point with five input/output ports. Figure 31 on page 89 is a basic representation of a node. The internal construction of each port of a node is shown in Figure 32 on page 90. Since a node does not have knowledge of the configuration of the network its receivers must always be enabled. Transmitters are enabled or disabled upon command. As a message is received, it will

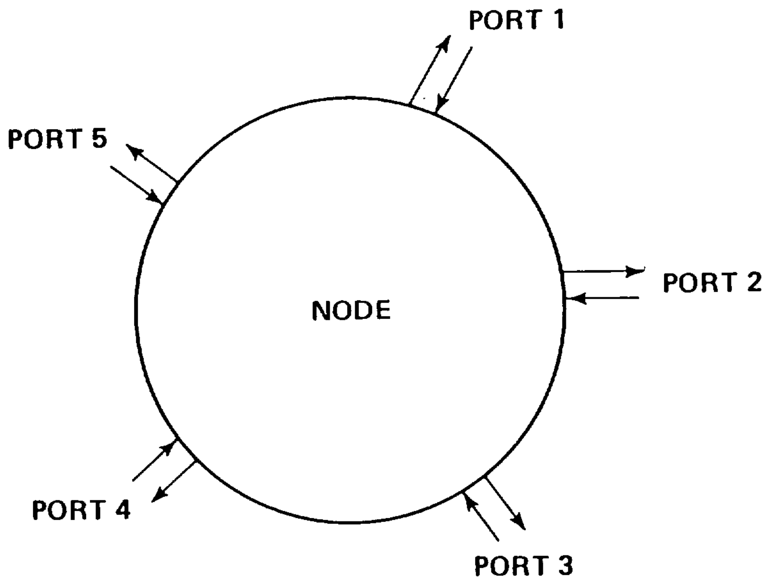

Figure 31. Input/Output Network Node

be regenerated and retransmitted on all enabled ports. At the same time, the message is decoded within the node. If the message is addressed to the node it will respond to the command embedded within the data. If the message is addressed elsewhere it will check for a valid transmission, latch observed error conditions and reset the receiver for the next transmission. When status is requested from the node, the observed errors will be transmitted within the status message.

Hardware will be used to inhibit selected ports from responding to or generating reconfiguration commands. Ports that are not so inhibited will be capable of receiving and processing reconfiguration commands whether active or not. Reconfiguration commands enable or disable selected port transmitters. Ports will be reconfigured whenever a command is received.

Some components are unique to a port (transmit/receive) and some are shared by all the ports (control). The following is a description of the basic components of a node.

3.5.1.4.1 Transmit/Receive Components

3.5.1.4.1.1 Receiver The receiver accepts the signal level on a link and converts it to the internal logic level of the port. The receiver also isolates the node from electrical failures of the link.

3.5.1.4.1.2 Protocol Decoder The protocol decoder accepts the serial data stream from the receiver, checks for protocol compliance and transmission induced errors. The errors are stored for transmission to GPCs when requested. It then synchronizes with the external transmission clock

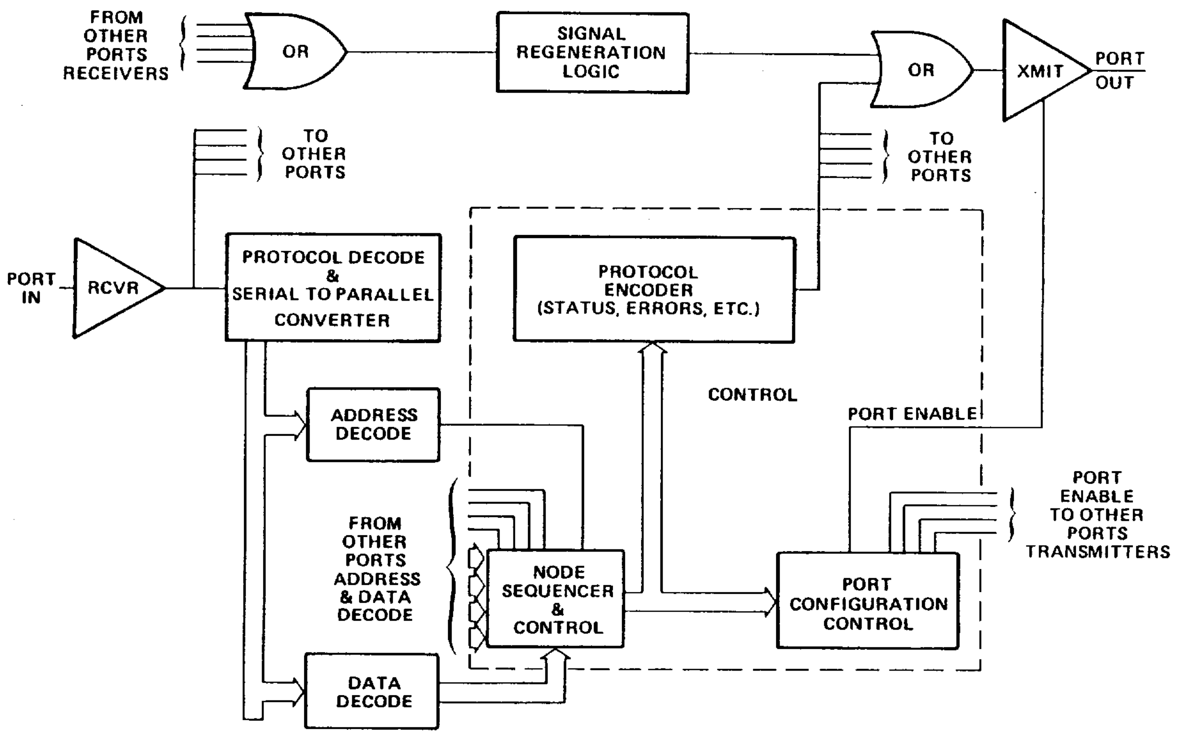
FROM OTHER PORTS RECEIVERS
OR
SIGNAL REGENERATION LOGIC
OR
XMIT
PORT OUT
TO OTHER PORTS
TO OTHER PORTS
PORT IN
RCVR
PROTOCOL DECODE & SERIAL TO PARALLEL CONVERTER
PROTOCOL ENCODER (STATUS, ERRORS, ETC.)
ADDRESS DECODE
CONTROL
PORT ENABLE
PORT ENABLE TO OTHER PORTS TRANSMITTERS
FROM OTHER PORTS ADDRESS & DATA DECODE
NODE SEQUENCER & CONTROL
PORT CONFIGURATION CONTROL
DATA DECODE

Figure 32. Input/Output Network Node Port

and assembles the message into parallel words. These parallel words are temporarily stored for the control section to examine.

3.5.1.4.1.3 Address Decoder The address decoder compares the address portion of the message with its own internally stored address. If the address matches the decoder signals the control section that there is a message waiting to be processed. If the address does not match the decoder section is cleared to wait for the next message.

3.5.1.4.1.4 Data Decoder If a message is addressed to the node the data decoder is used to extract the type of command from the data part of the message. The decoded command is sent to the control section.

3.5.1.4.1.5 Signal Regeneration Logic The signal regeneration logic is used to reconstruct the fidelity of the transmission. This is necessary because of the modification to a signal as it passes through circuit elements. After several transitions through circuit elements the transmission could appear to be modified. The input to the regeneration logic is the OR of all the ports. The output of the regeneration logic is applied to the input of the port transmitter.

3.5.1.4.1.6 Transmitter The transmitter converts the output of the protocol encoder or regeneration logic into the signaling levels used on the links. It can be enabled or disabled by the port configuration register.

3.5.1.4.2 Control Components

3.5.1.4.2.1 Node Sequencer and Control The node sequencer and control provides the central control for node operations. It accepts inputs from the address decoder indicating a message for the node has been received. It then decodes the command. There are two types of commands received by the node; 1)Reconfiguration commands and 2) Status requests. Configuration commands are passed to the port configuration register to control the ports. Status and error messages when requested by the processing site GPCs are passed to the protocol encoder. All control functions necessary to operate the encoder and decoder and the generation of timing signals for the node are generated within the sequencer.

3.5.1.4.2.2 Port Configuration Control The port configuration control accepts the decoded reconfiguration commands from the sequencer and enables or disables the individual port transmitters as commanded. The last command is stored until rewritten by the next command. This register can be read into the encoder if the status of the node is requested.

3.5.1.4.2.3 Protocol Encoder The protocol encoder receives the node response to status requests and encodes it into the link protocol. The output of the encoder is sent to the transmitters.

3.5.1.5 Mass Memory The AIPS mass memory is provided to support the POC system concept development and proof. While the mass memory is not required for the implementation of the AIPS concept it may be advantageous, in some applications, to have one.

The POC system mass memory consists of a generalized mass memory controller and a mass memory media. It is resident on a multiplexed bus which is in turn attached to those processing sites which are to make use of the mass memory. Contention for the multiplex mass memory bus shall be resolved with a Laning poll.

In the proof-of-concept system a generalized interface shall be used to allow for the inclusion of different types of memories. Memory types that are candidates for the proof of concept are RAM, UV PROM, EEROM, and Disc. The type of memory device chosen shall be transparent to the subsystem user. The mass memory will use error detection, error correcting, and/or memory replication for data protection. Figure 33 on page 92 is a block diagram of a generalized mass memory. The following is a description of the major elements of the mass memory.

3.5.1.5.1 Mass Memory Controller. The POC system mass memory controller shall provide a generalized interface between mass memory bus and the memory medium. The memory controller shall provide single fault masking for triplex data reception, fault detection for duplex data reception, and shall provide the capability for simplex data reception. Internally the memory controller shall be triply redundant. The generalized interface shall be capable of interfacing with various types of memory, such as,

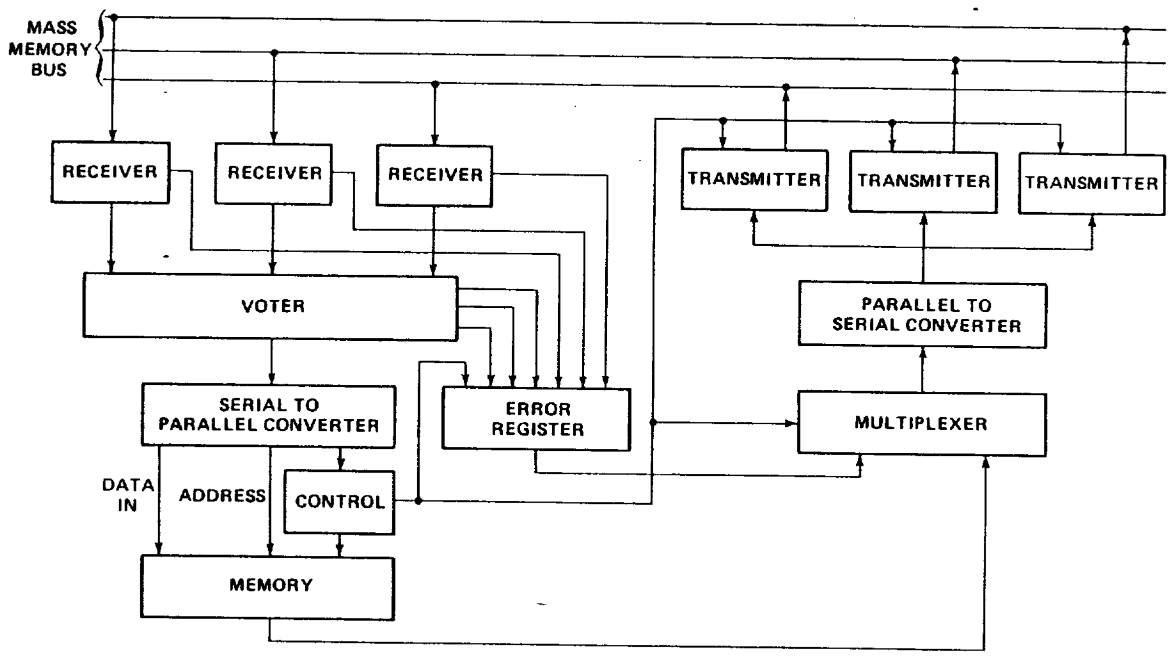

Figure 33. Mass Memory

disc, semiconductor and bubble. The following is a list of the mass memory controller elements.

(1) Receiver

The receiver accepts data from the bus, verifies conformance to the protocol, checks for transmission errors and formats the data for the memory. Formatting separates the data into address, data if a write, and command. Commands that are accepted include read, write and send status. The address, data and commands are then transferred to the verification section. Any errors that are detected at this level are stored in an error register which can be read by any element presently in control of the mass memory bus.

(2) Verification

The verification section does a compare on the address, data, and commands received from triplex or duplex elements. If the transmission is from a simplex element the mass memory must be conditioned not to use the voter and to accept the single string transmission. If a miscompare occurs on the transmission such that the verifier can not mask the error, use of the memory will be terminated and an error message returned. If a maskable error occurs, the transaction will be completed and the perceived error condition stored in the error register.

(3) Transmitter

The transmitter will receive the data from the memory, format and convert it to the signaling protocol for transmission to the requesting element.

(4) Control

The control section is responsible for examining the incoming service requests and generating the necessary commands and timing signals to execute the requests. It will generate the correct responses for read, write or request for status messages. The control section will disable the memory during polling and condition the verification section for simplex users. The write authority of simplex users will be limited to designated regions.

3.5.1.5.2 Memory Medium The memory section contains the storage devices. These storage devices as mentioned above can be RAM, EEPROM, ROM, UV PROM, Disc or magnetic tape. The choice of device can be deferred since the interface will appear to function identically with any of the above devices.

The POC mass memory recording medium will be selected to be appropriate for the POC system goals of concept development and demonstration within the laboratory environment.

3.5.1.6 Power Distribution The AIPS Power Distribution System (PDS) provides acceptable power to each of the AIPS processing sites. Power delivery shall continue in the presence of faults in sources, the distribution, or loads. The principal elements of the PDS are:

- Prime source conditioners,
- Distribution,
- Control.

3.5.1.6.1 Prime Source Conditioners The prime source conditioners convert power from the available sources to a form convenient for distribution and combination within AIPS. The prime sources may be aircraft power buses, fuel cells, solar cells, or the like in a given application. In the proof-of-concept system, the prime power sources will be building ac power, one utility and one preferably uninterruptible. The prime source conditioners will be specific to the application, but all must do the following:

(1) Convert prime power to medium voltage (28-48V) direct current for distribution.

(2) Protect the prime source from load faults:

(a) Overcurrent

(b) Overvoltage.

(3) Prevent propagation of source faults that may damage the loads or distribution system.

(4) Provide for remote shutdown of the source.

(5) Indicate source and load faults.

3.5.1.6.2 Distribution The prime source conditioners (above) and load combiners (below) serve to protect the sources, distribution, and subscribers from electrical faults. The distribution further provides a means of tolerating and circumventing physical damage that may occur at or between processing units.

As a minimum:

> A total failure event (one affecting all PDS connections) at a site or on a path between sites should not affect the delivery of power to other sites or allow damage to propagate beyond the area of initial occurrence.

Since this implies dispersal of distributed power buses, it will be necessary in any application to define the physical boundaries over which damage may take place as the result of a single event. For the proof-of-concept system, the boundaries will be:

(1) An enclosure, or

(2) A cable harness.

<u>Combination</u>

> A subscriber must draw power from a number of points within the distribution system so as to not be vulnerable to single failures. Normal combination works as follows:
> If any one power link to the combiner is carrying acceptable power, the combiner produces acceptable power.

3.5.1.6.3 Control The first level of control (protection) must be implemented to occur automatically or inherently within the elements of the PDS. Sources must be safe from defined load faults, and loads must be safe from defined source faults. Distribution must tolerate the stresses of maximum defined overloads due to failure of sources and loads, including failures of protective behavior.

The second level of control (remote) permits testing of the PDS and shedding of paths and elements to ascertain that other paths and elements would be usable in the event of failures. Load shedding may be used to terminate excessive power use by a failed subscriber or to remove a troublesome processing site for good.

Manual control shall be provided to allow maintenance of subscribers or portions of the PDS.

3.5.2 System Software

3.5.2.1 General Requirements

3.5.2.1.1 Programming Languages The AIPS architecture allows the simultaneous use of multiple high order languages.

For the proof-of-concept demonstration, the AIPS software will be programmed using a single high order language, Ada.

3.5.2.1.2 Function Partitioning Guidelines The following guidelines apply to the partitioning of functions among multiple GPCs. The software system will be designed to support these guidelines.

(1) There will be no inherent limitation on the number of functions that can be supported in a single GPC. The limitation will result from memory, cpu or application specific imposed restrictions.

(2) Related functions should be allocated to the same GPC so as to minimize the intercomputer information exchange traffic and the application transport lags.

(3) Alternate function partitions that are potential configurations resulting from function migration should be designed with consideration of performance margin requirements.

3.5.2.1.3 Software Structure The AIPS system software will place a minimum of constraints on the required structure of the application software. In particular, either or both of the two following application software structures will be possible.

3.5.2.1.3.1 Mission Phase Structure The mission phase structure organizes application software related to a specific mission phase into one software process. To illustrate, an application may wish to organize its Guidance and Flight Control software into say, launch, cruise, and landing phases. These phase organizations would be the 'Functions' from the viewpoint of the function migration requirements. This type of software structure can be thought of as a vertical organization because it includes elements of related 'application functions' (Guidance, Flight Control, etc.) in an executable process.

3.5.2.1.3.2 Application Function Structure The application function structure organizes all software for all mission phases related to an 'application function' into one software process. For example, all Guidance software related to say, launch, cruise, and landing phases would be organized into one executable process and the Flight Control software related to the same three phases would be organized into another executable process. These software organizations would be the 'Functions' from the viewpoint of the function migration requirements. This type of software structure can be thought of as a horizontal organization because it groups all code related to a particular 'application function' together in an executable process.

3.5.2.1.4 Growth and Change In order to achieve the AIPS change and growth goals, the software design should adhere to the following guidelines:

(1) The data structures that represent the physical characteristics of the system should be designed such that the addition of hardware, e.g., nodes, computer, or I/O devices, will not necessitate the reprogramming of software modules that assign or reference the structures.

(2) The above guideline should also be adhered to regarding the number of application functions the system is prepared to handle.

(3) The software design should adhere to a name scoping policy that avoids the problems of duplicate names arising when a new software function is added.

(4) It should be possible to treat the software for each computer as a separate entity from the link and load point of view. That is, a change in a software module that will execute on one computer should not necessitate a recompilation and relink of the software for another computer.

(5) The software protocols used for interfunction communication, intercomputer communication, and for sensor/effector I/O should be logically separated from the hardware protocols employed on the various communication media. This will insulate the software from changes in the hardware system.

Adherence to the above guidelines will require special attention on the part of both system software and application software designers and implementers. In particular:

Present and added users of I/O devices on the shared buses (global and regional) must be prepared to accept some delays due to both:

(1) contention over use of the bus by two or more computers, and

(2) queueing delays due to requests to use the same device concurrently (for devices that must be 'locked' for more than one bus transmission duration).

Guideline number (4) above will necessitate special care in the case of software for migratable functions. This software will exist in the link set (build) for each of the potential host computers. A change in such software must be followed by a relink of each host computer's software in order to effect the change in all potential execution sites. This relink of each affected GPC's software should be inforced by the software configuration management system.

3.5.2.1.5 Fault Tolerance The AIPS software system shall not disallow the use of various fault tolerant software approaches. The choice of which approach (or approaches) to utilize will be made by each application. Three well known techniques are the Recovery Block, the N-Version and the Backup Software approaches.

3.5.2.1.5.1 **Recovery Block** In this approach, there exists an ordered list of alternative implementations of the program. After the first (primary) alternative is executed an acceptance test is performed to determine the "appropriateness" of the output. If the test is successful, the output is used and no further alternatives are executed. If the test is unsuccessful, the next alternative is executed. If all alternatives have been executed without success, the recovery block terminates with an error code.

3.5.2.1.5.2 **N Version** In this approach, all alternative versions of the program are executed. When all versions have completed, a selection process determines the most appropriate output. This approach can be integrated with N-version hardware to provide for both hardware and software fault tolerance.

3.5.2.1.5.3 **Backup Software** In this approach, a separate backup version of the software is maintained in the event that an emergency situation makes execution of the primary version untenable. Once control is passed to the backup software, it retains control until remedial action is taken to correct the primary version. Typically, the backup software provides only those functions essential to safe operation. The function migration capability supports this approach by permitting the transfer of operation from one computing site to another which may host an alternate implementation of the function.

3.5.2.1.6 **Testability** The AIPS system software shall incorporate features to enhance the testability of the hardware and software.

3.5.2.1.6.1 **Fault Logging** The software shall record the occurrence of all detected hardware and software faults. The record shall identify the fault and the time of its occurrence.

3.5.2.1.6.2 **Resource Utilization** The system element operating systems shall monitor and record the usage duty cycle of their respective hardware resources. The primary purpose for this requirement is the analysis of system performance and the identification of throughput bottlenecks.

3.5.2.1.6.3 **Operating System Entry Trace** Each operating system will be capable of recording a chronological history of invoked entry points. The history trace will identify the operating system entry point, the time of the entry, and an identification of the invoking process. Certain software and hardware errors will cause the recording of the trace to stop. In addition, the operator will be able to start and stop the trace. Each GPC's history trace will be separately controlled.

3.5.2.1.7 **System Initialization and Restart** The system software will support the startup of the hardware elements in both their nominal, error free state as well as in most possible error conditions. The particular requirements follow.

- The system software shall support any power up sequence of AIPS hardware elements.

- The AIPS design philosophy relies on fault tolerant system elements. To be consistent with this philosophy, the power generation and distribution system should also be fault tolerant and provide protection against power transients. This implies that power transient restart protection is not required in the AIPS software. However, such protection can be provided on an application specific basis.

Power transient restart protection is not required for the Proof-of-Concept system.

3.5.2.2 Modes of Operation The AIPS system software operating modes are a composite of the modes of the devices (computers and gateways) that are functioning subscribers on the Intercomputer (IC) network. These modes are not dependent on the modes of operation of the application software.

The system software operating modes of the subscribers are:

(1) Startup and Initialization

(2) Normal

(3) Reconfiguration

(4) Fault Processing

(5) Test

These modes are not necessarily mutually exclusive, even within a single subscriber. For example, a uniprocessor could be reconfiguring one I/O network while simultaneously operating normally on another.

These modes are described in the following sections.

3.5.2.2.1 Startup and Initialization The startup and initialization mode includes the activities that the subscribers perform in the course of achieving a normal operating state. These activities are delineated in the following nominal startup sequence. Upon turn on, an IC network subscriber (computer or gateway) will do the following:

(1) perform internal initialization

(2) connect itself to its local I/O network (if any)

(3) connect itself to the IC network

(4) wait to receive a poll request that will allow it to inform the global of its status

If the computer is the designated global computer it will:

(1) determine the status of the global I/O network and establish a connection to all devices on the global I/O network

(2) determine the status of these I/O devices

(3) repeat the previous two steps for any regional I/O bus it is connected to

(4) set a value of system time

(5) determine the status of the IC network and establish the network so that it allows connection of all possible subscribers

(6) initiate the periodic poll of subscribers in order to establish their states - the global will accept as a member of the functioning set of subscribers any subscriber that responds to its poll message with a message that indicates that the subscriber is in functioning order

(7) from the set of functioning subscribers, the global will select a manager for each of the regional I/O networks

When a computer is established as a member of the functioning set it will complete its initialization:

(1) align its clock to the system time sent by the global

(2) connect itself to the global I/O network

(3) if the computer is a designated manager of a regional I/O network, it will establish that network

(a) connect itself to the regional I/O network

(b) determine the status of the network and establish a connection to each I/O device

(c) determine the status of each I/O device

(4) a computer that is not the manager for a regional network will connect to that network when it receives a permission message from the manager

A gateway completes its initialization by responding to the system poll message that it is capable of sending and receiving messages.

3.5.2.2.2 Normal The normal mode is the usual operating state - no fault processing is occurring nor is any reconfiguration, although self tests that are routinely executed are included. Two types of periodic messages are exchanged between the global computer and other IC network subscribers during normal processing:

(1) System Time Message

(2) Status Poll

The system time message is a broadcast transmission by the global computer. Each subscriber dedrifts its internal clocks with respect to the time value contained in the message. There is no explicit response to this message.

The status poll is a individual global to nonglobal subscriber transmission that includes an explicit response by the subscriber. The lack of a response indicates that the subscriber is off or otherwise isolated from the IC network. The subscriber response includes all the information the global computer requires to perform its system wide configuration management tasks. In addition, if the subscriber is a gateway, the response indicates whether or not a message from an adjoining network is pending.

3.5.2.2.3 **Reconfiguration** The reconfiguration mode includes the operations entailed in reconfiguring the system or system element. Reconfiguration is performed for a number of purposes such as:

spare cycling

restoration of fault free operation

function migration

fault identification

3.5.2.2.4 **Fault Processing** The fault processing mode is the operating state when a fault in a system element is being detected or logged. This includes the handling of software errors and exceptions.

3.5.2.2.5 **Test** The test mode is the operating state when the system element is performing a special test of itself or one of its components. Preflight and inflight checkout and diagnostic testing is included in this state.

3.5.2.3 Specific Requirements The specific functional software requirements are contained in this section. They are divided into two broad categories: Local Operating System functions, and Network Operating System functions as shown in Figure 34 on page 102.

Two local operating systems are included: a uniprocessor and a multiprocessor.

The Network Operating System categories include the intercomputer network functions, the global I/O network functions, the regional I/O network functions, and the mass memory functions.

A table of the system services that are provided vs the software functions is presented in Table 1 on page 103.

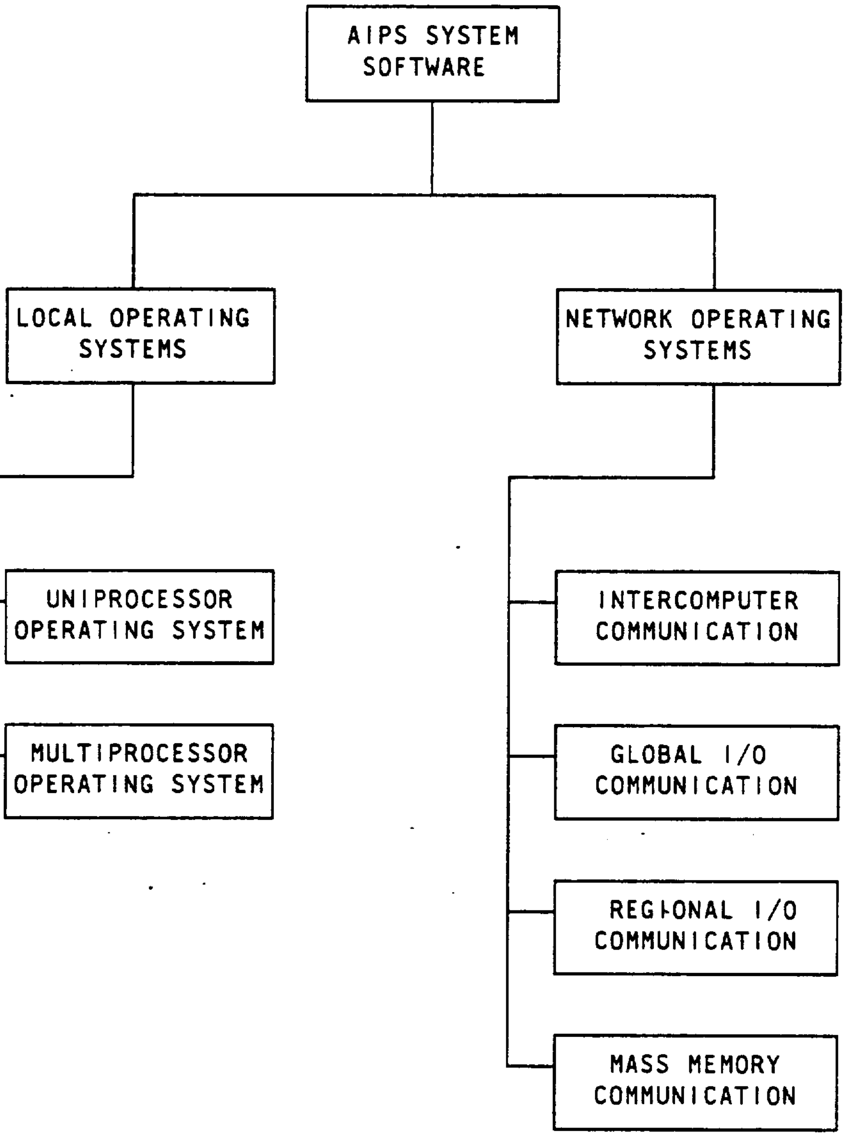
AIPS SYSTEM SOFTWARE
LOCAL OPERATING SYSTEMS
NETWORK OPERATING SYSTEMS
UNIPROCESSOR OPERATING SYSTEM
MULTIPROCESSOR OPERATING SYSTEM
INTERCOMPUTER COMMUNICATION
GLOBAL I/O COMMUNICATION
REGIONAL I/O COMMUNICATION
MASS MEMORY COMMUNICATION

Figure 34. Software Function Categories

Table 1. System Services versus Functions (Part 1 of 6)

FUNCTIONS	SYSTEM SERVICES					
	Local Comp. Mgmt.	IC Net Mgmt.	Non-Local I/O Mgmt.	Mass Memory Mgmt.	Time & File Mgmt.	Funct. Mgmt.
UNIPROCESSOR OPERATING SYSTEM						
Local Init.	X					
Local Restart	X					
Scheduling	X					
Dispatching	X					
Suspension	X					
Termination	X					
Intertask Communication	X					
Fault Detection	X					
Fault Identification	X					
Reconfiguration	X					
Synchronization	X					
Local Memory Mgmt	X					
Memory Read	X					
Memory Write	X					
Local I/O Bus Interface	X					
Local I/O Bus Mgr	X					
Exception Handler	X					

Table 1. System Services versus Functions (Part 2 of 6)

FUNCTIONS	SYSTEM SERVICES					
	Local Comp. Mgmt.	IC Net Mgmt.	Non-Local I/O Mgmt.	Mass Memory Mgmt.	Time & File Mgmt.	Funct. Mgmt.
Global Time Synch					X	
Terminate Function						X
Transmit Code/Data						X
Migrate Function						X
IC Net Interface		X				
Global I/O Net Interface			X			
Regional I/O Net Interface			X			
Normal Testing Support	X					
Special Testing Support	X					
MULTIPROCESSOR OPERATING SYSTEM						
Multiprocessor Init. & Restart	X					
Scheduler	X					
Dispatcher	X					
Local Communication	X					
Synchronization	X					

Table 1. System Services versus Functions (Part 3 of 6)

FUNCTIONS	SYSTEM SERVICES					
	Local Comp. Mgmt.	IC Net Mgmt.	Non-Local I/O Mgmt.	Mass Memory Mgmt.	Time & File Mgmt.	Funct. Mgmt.
Fault Detection	X					
Fault Identification	X					
Reconfiguration	X					
Self Tests	X					
Memory Management	X					
Memory Read	X					
Memory Write	X					
Local I/O Bus Interface	X					
Local I/O Bus Manager	X					
Exception Handler	X					
Global Time Synch					X	
Terminate Function						X
Transmit Code/Data						X
Migrate Function						X
IC Net Interface		X				
Global I/O Net Interface			X			

Table 1. System Services versus Functions (Part 4 of 6)

FUNCTIONS	SYSTEM SERVICES					
	Local Comp. Mgmt.	IC Net Mgmt.	Non-Local I/O Mgmt.	Mass Memory Mgmt.	Time & File Mgmt.	Funct. Mgmt.
Regional I/O Net Interface			X			
Normal Testing Support	X					
Special Testing Support	X					
NETWORK OPERATING SYSTEM						
IC net fault identification		X				
Subscriber Poll Response		X				
IC Net Init.		X				
Context Manager		X				
Reconfigure IC Net		X				
IC Net Communication		X				
IC Net Manager		X				
Initial Program Load						X
Applications Program Load						X
Periodic Subscriber Poll		X				

Table 1. System Services versus Functions (Part 5 of 6)

FUNCTIONS	SYSTEM SERVICES					
	Local Comp. Mgmt.	IC Net Mgmt.	Non-Local I/O Mgmt.	Mass Memory Mgmt.	Time & File Mgmt.	Funct. Mgmt.
Function Migration						X
Broadcast System Time					X	
Set System Time					X	
System Time Source FDIR					X	
Signal Message Failure			X			
Initialize Global I/O Device			X			
Transmit To Global I/O Device			X			
Manage Contention For Global I/O			X			
Reconfigure Global I/O Net			X			
Check Status Of Global I/O Net			X			
Check Idle Global I/O Device			X			
Signal Message Failure			X			
Initialize Regional Net			X			
Initialize Regional Device			X			

Table 1. System Services versus Functions (Part 6 of 6)

FUNCTIONS	SYSTEM SERVICES					
	Local Comp. Mgmt.	IC Net Mgmt.	Non-Local I/O Mgmt.	Mass Memory Mgmt.	Time & File Mgmt.	Funct. Mgmt.
Transmit To Regional Device			X			
Manage Contention For Regional I/O			X			
Reconfigure Regional I/O Net			X			
Check Status Of Regional I/O			X			
Check Idle Regional Device			X			
File Management				X	X	
Mass Memory Read And Write				X	X	

3.5.2.3.1 Local Operating Systems The local operating systems of the general purpose computers (uniprocessor and multiprocessor) provide the services discussed in section 3.1.3.1, Local Computer Management.

3.5.2.3.1.1 Uniprocessor Operating System The uniprocessor operating system provides the services necessary for the operation of the fault tolerant processor. Those services include local management functions such as initialization and restart, local configuration management, scheduling and dispatching tasks, and communication between tasks. It also provides the interfaces to the network operating systems. See Figure 35 on page 110.

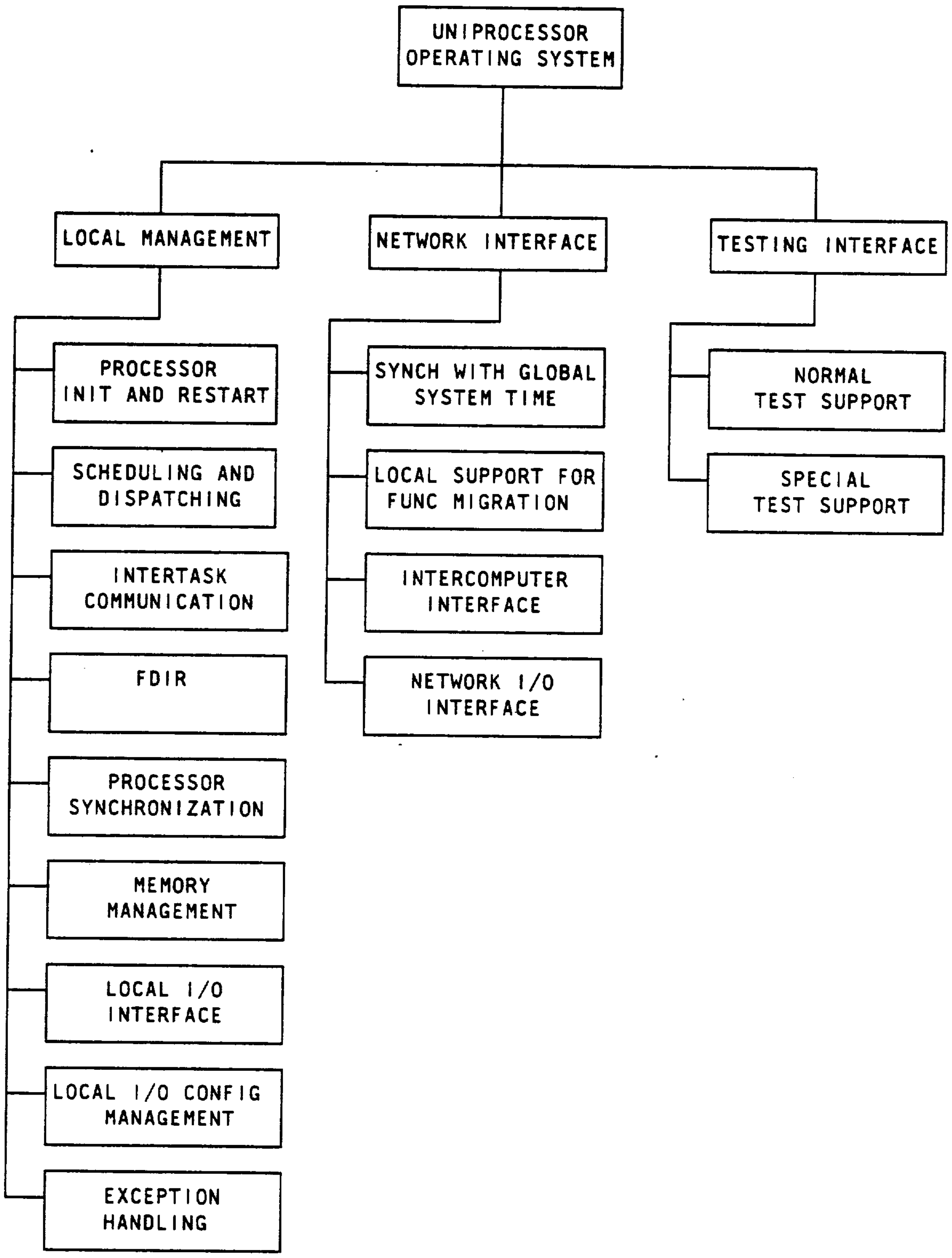

Figure 35. Uniprocessor Operating System Functions

3.5.2.3.1.1.1 Local Management

3.5.2.3.1.1.1.1 Processor Initialization and Restart

Function Name: Local Uniprocessor Initialization

Requirements Source: CSDL

Modes in Which Function Required:

Startup

Initiation and Termination Events:

Initiation: Local power turn on, operator request
Termination: Initiation of normal/test operating mode

Functional Description of Inputs:

The inputs to this function shall include:

- initial system configuration/status tables
- initial local configuration/status tables
- initial operating system program load
- initial application program load.

Functional Description of Outputs:

The outputs from this function shall include:

- updated system configuration/status tables
- updated local configuration/status tables.

Description of Function:

This function shall be responsible for bringing a local uniprocessor GPC from a cold startup to the normal/test operating mode and may include the following actions:

- obtain initial operating system program load from Read Only Memory or mass memory
- initialize local operating system database
- synchronize local redundant computational and I/O processors
- determine status of and configure local redundant processors
- perform system startup procedures to connect with intercomputer and shared I/O networks

- determine status of any memory mapped I/O devices
- initialize local I/O bus:

  determine status of local I/O bus

  if network, establish connections to each I/O device

  determine status of each I/O device
- obtain initial application program load from either resident Read Only Memory, mass memory or another processing site.
- schedule initial applications tasks for execution.

Comments:

The specific actions performed by this function shall be determined by the hardware architecture of the particular AIPS application.

Function Name: Local Uniprocessor Restart

Requirements Source: CSDL, AIPS-83-50

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Power transient, software error conditions
Termination: Resumption of normal/test operating mode

Functional Description of Inputs:

The inputs to this function shall include:

- current system configuration/status tables
- current local configuration status tables
- current operating system program load
- current application program load

Functional Description of Outputs:

The outputs from this function shall include:

- updated system configuration/status tables
- updated local configuration/status tables.

Description of Function:

This function shall attempt to gracefully return a local uniprocessor GPC to a normal/test operating mode following the occurrence of certain hardware/software faults. The function may include the following actions:

- obtain current operating system program load from resident Read Only Memory or mass memory
- initialize local operating system database
- synchronize local redundant computational and I/O processors
- determine status of and configure local redundant processors
- perform system startup procedures to reconnect with intercomputer and shared I/O networks
- reconnect with local I/O bus

Comments:

The specific actions performed by this function shall be determined by the hardware architecture of the particular AIPS application.

If, due to the occurrence of certain failure modes, the local uniprocessor GPC is unable to gracefully execute this function, the full Local Uniprocessor Initialization function shall be initiated instead.

3.5.2.3.1.1.1.2 Scheduling and Dispatching

Function Name: Task Scheduling

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Task schedule call
Termination: Task ready for dispatch

Functional Description of Inputs:

The inputs to this function shall include:

- task identifier
- scheduling criteria:

  priority

  execution time

  event identifier
- current schedule queues.

Functional Description of Outputs:

The outputs from this function shall include:

- updated schedule queue.

Description of Function:

This function shall schedule tasks in a multitasking environment that shall permit both synchronous and asynchronous operation.

Tasks may be scheduled by other tasks or by the operating system (system tasks) according to three criteria: priority, time or event occurrence.

A priority task shall be either dispatched immediately or queued as a ready task depending on whether it has a higher or lower priority than the currently active task.

Time dependent tasks may be either one time or cyclic. Time tasks shall be queued according to execution time. At execution time, tasks shall be dispatched as priority tasks with cyclic time tasks being requeued according to the updated execution time.

Event dependent tasks may be either one time or cyclic. Event tasks shall be queued according to an event identifier (event number). Event occurrence shall be signalled via an operating system primitive. At event occurrence, tasks shall be dispatched as priority tasks with cyclic event tasks being requeued according to event number.

Comments:

None.

Function Name: Task Dispatching

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Task ready for execution
Termination: Task execution initiated

Functional Description of Inputs:

The input to this function shall include:

- current schedule queue.

Functional Description of Outputs:

The output from this function shall include:

- updated schedule queue
- task watchdog timer.

Description of Function:

This task shall activate a task ready for execution. Associated with each defined task shall be a numerical priority. Task dispatching or activation shall be priority driven with tasks ready for dispatch queued according to priority. A task shall be activated when it is the highest priority ready task. An active task shall be interrupted by a ready task of higher priority. The interrupted task shall be queued according to priority and dispatched again when it has the highest priority.

A watchdog timer shall be activated prior to task dispatching to protect against exclusive CPU takeover by a task.

Comments:

None.

Function Name: Task Suspension

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Task suspension call
Termination: Task reactivated

Functional Description of Inputs:

The inputs to this function shall include:

- task identifier
- task reactivation criteria:

  reactivation time

  event identifier
- current schedule queues.

Functional Description of Outputs:

The outputs from this function shall include:

- updated schedule queues.

Description of Function:

This function shall suspend a currently active task. Tasks may be suspended (by themselves or by the operating system) pending a time delay or an event occurrence. The suspended task shall be queued as a time or an event task and dispatched as a priority task whenever the specified time delay or event occurs.

Comments:

None.

Function Name: Task Termination

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Task termination call
Termination: Task removed from schedule queues

Functional Description of Inputs:

The inputs to this function shall include:

- task identifier
- current schedule queues.

Functional Description of Outputs:

The outputs from this function shall include:

- updated schedule queues.

Description of Function:

This function shall attempt to gracefully terminate execution of a task. Upon completion of normal task execution, the completed task shall be deactivated and the highest priority ready task dispatched. A task that has been scheduled (by priority, time or event) may be cancelled by another task or the operating system before being dispatched. An active, interrupted or suspended task shall be capable of being aborted by another task or the operating system.

Comments:

Although task aborts shall be supported by this function, specific programming guidelines indicating under what conditions such aborts shall be permitted will be developed for each AIPS application.

3.5.2.3.1.1.1.3 Intertask Communication

Function Name: Local Intertask Communication

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Software call
Termination: Completion of communication action

Functional Description of Inputs:

The inputs to this function may include:

- communicating task identifiers
- parameters to be passed
- pointers to shared memory areas (including mailboxes)
- current schedule queues.

Functional Description of Outputs:

The outputs from this function may include:

- parameters passed
- updated schedule queues.

Description of Function:

This function shall support the following types of communication between concurrently executing tasks:

Task synchronization	The execution of all or part of a task is dependent on the execution of all or part of another task.
Parameter passing	Required parameters are passed back and forth between tasks.
Shared data lockout	Access to a set of shared data is restricted to one task at a time.

Comments:

None.

3.5.2.3.1.1.1.4 FDIR

Function Name: Fault Detection

Requirements Source: CSDL, AIPS-83-50

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Computer turn on
Termination: Computer turn off

Functional Description of Inputs:

The inputs to this function shall include:

- CPU, memory and data exchange hardware self test results
- hardware error latch registers
- normal I/O data.

Functional Description of Outputs:

The outputs from this function shall include:

- hardware error latch registers.

Description of Function:

This function shall detect hardware faults within redundant channels of the Fault Tolerant Processor by periodic exchange of data between the cross strapped computational and I/O processors. Data exchanged may include results of CPU, memory and data exchange hardware self test routines, hardware error latch registers and, in the case of I/O processors, I/O data during normal voting and source congruency exchanges. Data miscompares will be indicated in hardware error latches and shall represent a detected fault. For a simplex processor, fault detection shall be attempted via background self test routines.

Comments:

Fault detection through data comparison requires redundant processor execution in a tightly coupled, instruction synchronized mode.

Function Name: Fault Identification

Requirements Source: CSDL, AIPS-83-50

Modes in Which Function Required:

Fault Processing

Initiation and Termination Events:

Initiation: Detection of fault
Termination: Identification of fault

Functional Description of Inputs:

The inputs to this function shall include:

- hardware error latch registers.

Functional Description of Outputs:

The outputs from this function shall include:

- fault identification indicator.

Description of Function:

This function shall identify hardware faults within redundant channels by periodic software analysis of exchanged hardware error latch registers. Identification shall be attempted as to location and duration ('soft' or transient, 'hard'). The occurrence of a predetermined number of transient or 'soft' faults within a fixed time interval shall be considered a 'hard' fault. Fault identification in a simplex processor shall be attempted by analysis of self test results.

Comments:

None.

Function Name: Processor Reconfiguration

Requirements Source: CSDL, AIPS-83-50

Modes in Which Function Required:

Startup
Reconfiguration

Initiation and Termination Events:

Initiation: Fault identification
Termination: Processor reconfiguration completed

Functional Description of Inputs:

The inputs to this function shall include:

- fault identification indicator
- current local configuration/status tables.

Functional Description of Outputs:

The outputs from this function shall include:

- updated local configuration/status tables
- processor mask registers
- updated status message to global computer.
- updated fault log.

Description of Function:

This function shall reconfigure the Fault Tolerant Processor (FTP) upon identification of a detected fault. Successful identification of a faulty component in a redundant FTP channel shall result in the effected component being masked out of any further FTP activities (data exchanges, voting, I/O etc.). Due to the cross strapping of processors, the failure of one component shall not necessarily result in the loss of the entire channel; e.g. a failed computational processor shall not result in the masking of its corresponding I/O processor's I/O activity.

Periodically a faulty component shall be tested to determine if the fault is persisting. Reconfiguring a previously faulty component on line shall be attempted after fault free operation for a predetermined time interval.

Reconfiguration shall be prohibited in the case of unsuccessful fault identification in a duplex or simplex processor. The global computer shall be notified of the unreliability of the processor's output.

All faults detected and changes of processor status shall be logged for transmission to the global computer for possible system function migration.

Comments:

Depending on the redundancy requirements of the particular functions being implemented at a specific FTP processing site, termination of task execution may be required upon reconfiguration.

3.5.2.3.1.1.1.5 Processor Synchronization

Function Name: Processor Synchronization

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: power turn on, periodic demand
Termination: redundant processors synchronized

Functional Description of Inputs:

The inputs to this function shall include:

- initial local configuration/status tables.

Functional Description of Outputs:

The outputs from this function shall include:

- updated local configuration/status tables.

Description of Function:

This function shall tightly synchronize the redundant processors (computational and I/O) in a Fault Tolerant Processor to the instruction level. Synchronization shall be accomplished utilizing the data exchange cross strapping between redundant processors. Failure of a processor to achieve synchronization with its neighbor(s) shall be interpreted as a fault in the indicated processor with resultant downgrading of that processing site's redundancy level.

Synchronization shall be attempted at power on ('cold' start) and at periodic intervals to ensure continued synchronization and to attempt recapture of any 'lost' processor(s).

Comments:

None.

3.5.2.3.1.1.1.6 Memory Management

Function Name: Local Memory Management

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Task initiation, user request
Termination: Task completion, user request

Functional Description of Inputs:

The inputs to this function shall include:

- memory allocation request
- pointer to available memory storage area.

Functional Description of Outputs:

The outputs from this function shall include:

- updated pointer to available memory storage area
- software exception.

Description of Function:

This function shall dynamically allocate and deallocate local memory for executing tasks. A software exception shall be raised if the required memory allocation exceeds the available storage size.

Comments:

None.

Function Name: Uniprocessor Memory Read

Requirements Source: CSDL

Modes in Which Function Required:

Normal

Initiation and Termination Events:

Initiation: on demand

Functional Description of Inputs:

The inputs to this function are the addresses of the local memory locations to be read.

Functional Description of Outputs:

The outputs of this function are the contents of the local memory locations.

Description of Function:

This function responds to a request for the value of the contents of specified local memory locations.

Comments:

None.

Function Name: Uniprocessor Memory Write

Requirements Source: CSDL

Modes in Which Function Required:

Normal

Initiation and Termination Events:

Initiation: on demand

Functional Description of Inputs:

The inputs to this function are the addresses of the local memory locations to be written into and the values to be stored.

Functional Description of Outputs:

The outputs of this function are the altered contents of the local memory locations.

Description of Function:

This function responds to a request to alter the contents of specified local memory locations.

Comments:

None.

3.5.2.3.1.1.1.7 Local I/O Interface

Function Name: Local I/O Bus Interface

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On demand
Termination: Local I/O bus transaction completed

Functional Description of Inputs:

The inputs to this function shall include:

- formatted command/data message for local I/O device.

Functional Description of Outputs:

The outputs from this function shall include:

- response from local I/O device
- transaction status.

Description of Function:

This function shall transmit messages to and receive responses from local I/O devices as a service for the local I/O bus manager of the local operating system. Source congruency shall be applied to messages prior to transmission and to responses after receipt. Upon completion, the status of the transaction shall be returned to the local I/O bus manager.

Comments:

None.

3.5.2.3.1.1.1.8 Local I/O Configuration Management

Function Name: Local I/O Bus Manager

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Computer turn on
Termination: Computer turn off

Functional Description of Inputs:

The inputs to this function shall include:

- current local I/O bus configuration/status tables
- commands/data for I/O device on local I/O bus.

Functional Description of Outputs:

The outputs from this function shall include:

- updated local I/O bus configuration/status tables
- reconfiguration commands to the local I/O bus (if a network)
- data from I/O device on local I/O bus.

Description of Function:

This function shall be responsible for:

- servicing requests for initiating transactions to local I/O devices
- servicing responses from local I/O devices
- resolving contentions for local I/O devices
- detection of faulty I/O devices
- in the case of a local I/O network, detection of node/link faults, network reconfiguration and periodic incorporation of inactive links.

Comments:

None.

3.5.2.3.1.1.1.9 Software Exception Handler

Function Name: Local Operating System Software Exception Handler

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Detection of software exception.
Termination: Return to application task.

Functional Description of Inputs:

The inputs to this function shall include:

- exception identifier
- error log
- task identifier

Functional Description of Outputs:

The outputs from this function shall include:

- updated error log

Description of Function:

Upon a software exception, this function shall log the exception and return to the application specified software error handler. If there is no application specified error handler, the function will perform a standard error fix up.

Comments:

None.

3.5.2.3.1.1.2 System Interfaces

3.5.2.3.1.1.2.1 Synchronization with Global System Time

Function Name: Global Time Synchronization

Requirements Source: CSDL, AIPS-83-50

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Receipt of system time message
Termination: Local time synchronized with system time

Functional Description of Inputs:

The inputs to this function shall include:

- system time message from global computer.

Functional Description of Outputs:

The outputs from this function shall include:

- updated local time value
- time drift correction.

Description of Function:

This function shall synchronize local processor time with the broadcast system time. Upon receipt of the periodic system time message from the global computer, the enclosed time value will be compensated by known biases and the resultant value used to dedrift the local time.

Comments:

None.

3.5.2.3.1.1.2.2 Local Support for Function Migration

Function Name: Terminate Function Execution

Requirements Source: CSDL, AIPS-83-50

Modes in Which Function Required:

Reconfiguration

Initiation and Termination Events:

Initiation: Command from global computer to terminate function
Termination: Function execution terminated

Functional Description of Inputs:

The inputs to this function shall include:

- message from global computer to terminate specified function.

Functional Description of Outputs:

The outputs from this function shall include:

- message to global computer indicating termination status
- updated local schedule queues.

Description of Function:

Upon command of the global computer, an attempt shall be made to 'gracefully' abort a currently executing function at an application specified checkpoint. If a checkpoint is not reached within a predetermined time interval, the function shall be terminated immediately. All open data files shall be closed and any locked data shall be unlocked. All tasks relating to the terminated function shall be deleted from the local system queues. A terminated function shall be allowed to execute again at this processing site only when it has been reactivated by the global computer (e.g.as a result of a future function migration). The status of the function termination shall be transmitted to the global computer.

Comments:

None.

Function Name: Transmit Code/Data

Requirements Source: CSDL, AIPS-83-50

Modes in Which Function Required:

Reconfiguration

Initiation and Termination Events:

Initiation: Receipt of code/data transmission request
Termination: Code/data transmitted

Functional Description of Inputs:

The inputs to this function shall include:

- transmission request containing:

  code/data identifier

  destination processing site identifier.

Functional Description of Outputs:

The outputs from this function shall include:

- message to destination processing site containing required code/data.

Description of Function:

Upon receipt of the code/data transmission request, the indicated code/data (including checkpoint data) resident at this processing site shall be transferred to the destination processing site. Transmission may be to mass memory to be retrieved by the destination processing site or, alternately, over the intercomputer network directly to the destination site.

Comments:

None.

Function Name: Migrate Function

Requirements Source: CSDL, AIPS-83-50

Modes in Which Function Required:

Reconfiguration

Initiation and Termination Events:

Initiation: Function migrate command from global computer
Termination: Function migrated and activated

Functional Description of Inputs:

The inputs to this function may include:

- function migrate message from global computer including:

  function identifier

  code/data source identifier

- message from source processing site containing required function code/data

- local schedule queues.

Functional Description of Outputs:

The outputs from this function may include:

- code/data transmission request to source processing site including:

  function code/data identifier

  destination processing site identifier

- message to global computer giving status of function migration

- updated local schedule queues.

Description of Function:

Upon receipt of the function migrate command from the global computer, the indicated function's code/data shall be obtained from mass memory directly or from another processing site utilizing the intercomputer network or mass memory. In either case, the received code/data shall be prepared for execution/use at this processing site (virtual memory page registers initialized etc.).

The indicated function shall be scheduled for execution and activated at its initial start address or at a restart checkpoint. The status of the function migration and activation shall be reported to the global computer.

Comments:

None.

3.5.2.3.1.1.2.3 Intercomputer Interface

Function Name: Intercomputer (IC) Network Interface

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On demand
Termination: IC network transaction completed

Functional Description of Inputs:

The inputs to this function shall include:

- formatted message for transmission on IC network.

Functional Description of Outputs:

The outputs from this function shall include:

- formatted message received from IC network
- IC transaction status.

Description of Function:

This function shall transmit and receive messages from the IC network as a service for the IC network functions of the network operating system. Output messages on redundant channels shall be voted in hardware prior to transmission. The status of input messages shall be determined from hardware error latches and the results passed along to the user IC network function.

Comments:

None.

3.5.2.3.1.1.2.4 Network I/O Interface

Function Name: Global I/O Network Interface

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On demand
Termination: Global I/O network transaction completed

Functional Description of Inputs:

The inputs to this function shall include:

- formatted command/data message for global I/O device.

Functional Description of Outputs:

The outputs from this function shall include:

- response from global I/O device
- transaction status.

Description of Function:

This function shall transmit messages to and receive responses from global I/O devices as a service for the global I/O network functions of the network operating system. Source congruency shall be applied to messages prior to transmission and to responses after receipt. Upon completion, the status of the transaction shall be returned to the global I/O network function.

Comments:

None.

Function Name: Regional I/O Network Interface

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On demand
Termination: Regional I/O network transaction completed

Functional Description of Inputs:

The inputs to this function shall include:

- formatted command/data message for regional I/O device.

Functional Description of Outputs:

The outputs from this function shall include:

- response from regional I/O device
- transaction status.

Description of Function:

This function shall transmit messages to and receive responses from regional I/O devices as a service for the regional I/O network functions of the network operating system. Source congruency shall be applied to messages prior to transmission and to responses after receipt. Upon completion, the status of the transaction shall be returned to the regional I/O network function.

Comments:

None.

3.5.2.3.1.1.3 Testing Interfaces

3.5.2.3.1.1.3.1 Normal Test support

Function Name: Normal Test Support

Requirements Source: CSDL

Modes in Which Function Required:

Normal

Initiation and Termination Events:

Initiation: GPC startup
Termination: GPC shutdown

Functional Description of Inputs:

The inputs to this function include faults to be logged, operating system entry identifiers, and watchdog timer update commands.

Functional Description of Outputs:

The outputs of this function is the specified action.

Description of Function:

This function provides the normal testing support for the processor. These functions include: the logging of all hardware and software faults, the maintenence of an operating entry trace, and the maintenence of a watchdog timer.

Comments:

None.

3.5.2.3.1.1.3.2 Special Test support

Function Name: Special Test Support

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: GPC startup
Termination: GPC shutdown

Functional Description of Inputs:

The inputs to this function include processor halt commands, and commands to simulate faults.

Functional Description of Outputs:

The outputs of this function is the specified action.

Description of Function:

This function provides the special testing support for the processor. These functions include halting the processor upon the detection of specified conditions and simulating specified faults for special testing operations.

Comments:

None.

3.5.2.3.1.2 Multiprocessor Operating System The multiprocessor operating system provides the services necessary for the operation of the fault tolerant multiprocessor. Those services include local management functions such as initialization and restart, local configuration management, scheduling and dispatching tasks, and communication between tasks. It also provides the interfaces to the network operating systems. See Figure 36 on page 143.

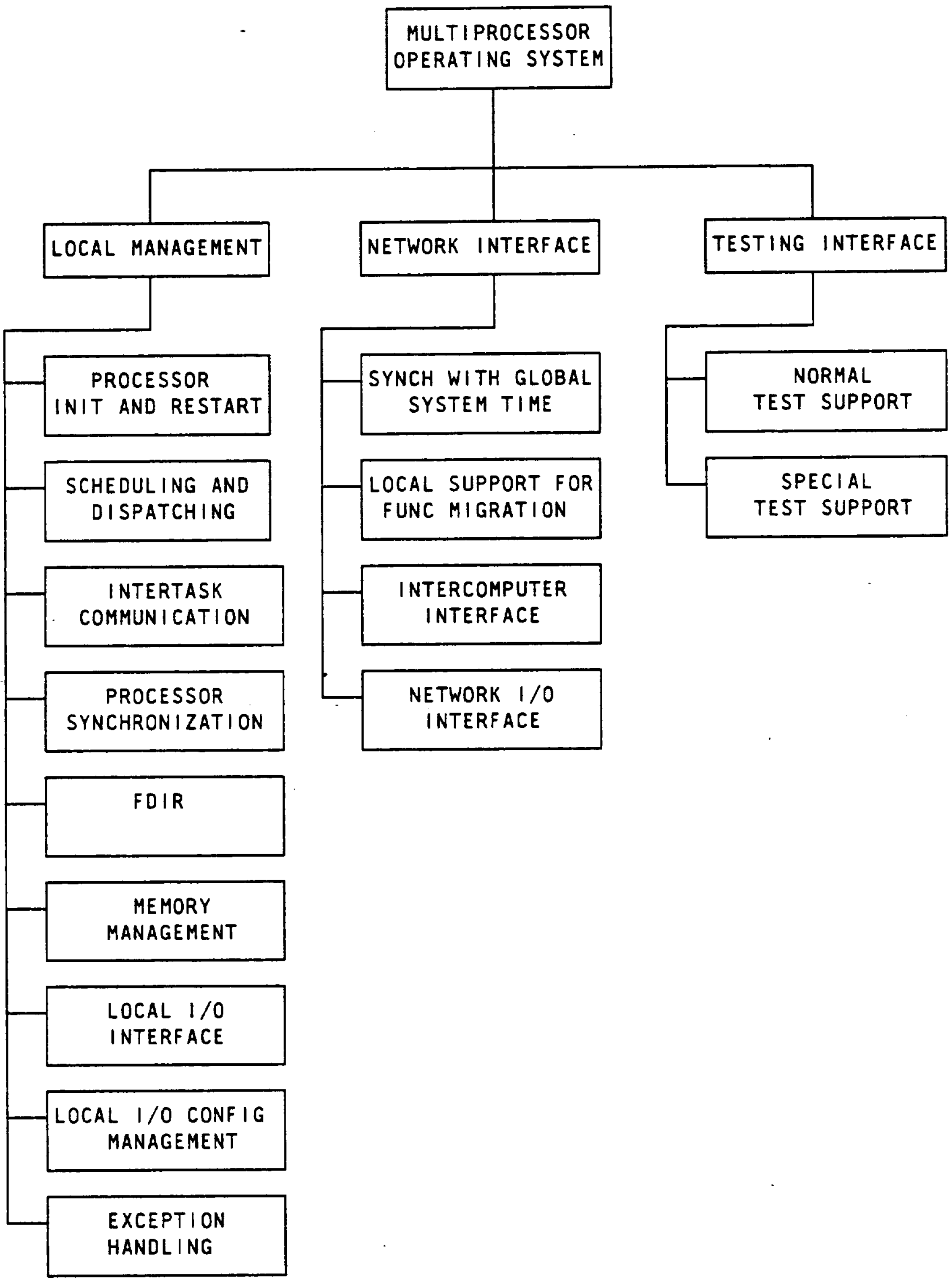

Figure 36. Multiprocessor Operating System Functions

3.5.2.3.1.2.1 Local Management

3.5.2.3.1.2.1.1 Processor Initialization and Restart

Function Name: Multiprocessor Initialization and Restart

Requirements Source: CSDL

Modes in Which Function Required:

Startup,Restart

Initiation and Termination Events:

Initiation: Multiprocessor Startup
Termination: Multiprocessor Configuration Complete

Functional Description of Inputs:

The inputs to this function include AIPS system configuration tables and local system configuration tables.

Functional Description of Outputs:

The outputs of this function include updated AIPS system configuration tables and undated local system configuration tables.

Description of Function:

This function performs two services: one, to bootstrap the system initially, also called the "cold start"; and two, to restart the system after a power interruption, also called the "hot start".
Cold start. In the case of the cold start, neither the local system memory nor the control registers will contain useful information. The system memory will be bootstrapped from mass memory, or another external device. The multiprocessor system configuration tables and the local copy of the AIPS system configuration tables in the multiprocessor's shared memory will be initialized and the rest of the system start functions will be the same as in the hot start.
Hot start. The multiprocessor system will be restarted in the case of a power failure. The system memory and the system control unit registers shall be non volatile. Cache RAM will be volatile. The function may perform the following restart actions:

- Reset hardware control registers
- Clear interval timers
- Clear all interrupts
- Initialize cache RAM
- Synchronize with other members of triad

- Read local system configuration tables
- Bring the multiprocessor system to system configuration: including configuring memory, processors, buses, clocks, and local I/O network
- Read AIPS system configuration tables
- Perform procedures to connect with IC and shared I/O networks

Comments:

The restart function should reside in non volatile PROM.

3.5.2.3.1.2.1.2 Scheduling

Function Name: Multiprocessor Scheduler

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Operating system, application task schedule call
Termination: Task is placed on appropriate queue

Functional Description of Inputs:

The inputs to this function include the task identification, queues, priority of the task, event occurrence, and the time to schedule.

Functional Description of Outputs:

The outputs include a signal to another active triad to schedule the next time dependent task, an updated ready queue and activation of the dispatcher.

Description of Function:

The operating system shall manage task execution in a multiprogram multiprocessor environment. All parts of the multiprocessor operating system shall be able to run on any of the active processor triads.

Time dependent tasks may be either one time or cyclic. Time tasks shall be queued according to execution time. At execution time, tasks shall be put on the ready queue and dispatched according to priority. Time dependent tasks may be scheduled by any of the active triads. If more than one triad is active at scheduling time, the responsible triad will kick another active triad to be responsible for scheduling the next task in the time dependent queue. Cyclic time tasks will be requeued according to time to schedule.

Event dependent tasks may be either one time or cyclic. They shall be queued in order of an event identifier. At event occurrence, the task shall be scheduled by the triad that sees the event first. Cyclic event tasks shall be requeued in order of event number.

A priority task shall be either dispatched immediately or queued as a ready task depending on whether it has a higher priority than any of the tasks presently running. If it has a higher priority, the task with the least priority of the active tasks shall be interrupted.

Comments:

None.

3.5.2.3.1.2.1.3 Dispatching

Function Name: Multiprocessor Dispatcher

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Scheduler, Task completion signal
Termination: Context switch completed

Functional Description of Inputs:

The inputs to this function include the priority of the task to be dispatched, the appropriate pointers to that task and the task to be interrupted.

Functional Description of Outputs:

The outputs include appropriate pointers to an interrupted task, updated queues and a task watchdog timer.

Description of Function:

Associated with each defined task shall be a numerical priority. Task dispatching or activation shall be priority driven with tasks ready for dispatch being queued according to priority. A task shall be activated when it is the highest priority ready task. An active task shall be interrupted by a ready task of higher priority. Of the active tasks running, the one with the least priority shall be the interrupted one. The interrupted task shall be queued according to priority and dispatched again when it has the highest priority.

The dispatcher task shall run on any of the active processor triads. It will be activated by the scheduler whenever a task is scheduled that has a higher priority than any presently running. It will also be activated when any presently running task reaches normal completion.

A watchdog timer for each triad processor shall be activated at task dispatch time to protect against CPU takeover by an application task.

Comments:

None.

3.5.2.3.1.2.1.4 Local Intertask Communication

Function Name: Multiprocessor Local Communication

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Operating system activation
Termination: Communication completed

Functional Description of Inputs:

The inputs to this function may include pointer to parameters to be passed, pointers to shared memory (mailboxes), communicating task identifier.

Functional Description of Outputs:

The outputs of this function may include parameters passed, and messages to other triads.

Description of Function:

Intertask Communication. This function shall support the following types of communication between tasks running on the same processor triad and/or tasks running on different processor triads:

Task synchronization	The execution of all or part of a task is dependent on the execution all or part of another task.
Parameter passing	Required parameters are passed back and forth between tasks.
Shared memory lockout	A task will be allowed to lock portions of the multiprocessor's shared memory.

Operating System Communication. The multiprocessor operating system shall be able to directly communicate to an active processor triad by means of hardware IPC (interprocessor triad communication) registers. Writing to these registers can cause an IPC interrupt. There will be an IPC interrupt handler whose function it is to interrupt the task presently running in that triad, interpret the communication from the operating system, and take the appropriate action.

Comments:

None.

3.5.2.3.1.2.1.5 Processor Synchronization

Function Name: Synchronization

Requirements Source: CSDL

Modes in Which Function Required:

Startup
Fault Processing
Reconfiguration

Initiation and Termination Events:

Initiation: call by restart or reconfiguration function
Termination: triad is synchronized

Functional Description of Inputs:

Functional Description of Outputs:

Description of Function:

In order for the multiprocessor to maintain the appropriate level of reliability, the CPs and IOPs of the system must form triads that work together in tight synchronism. This function will synchronize the processors.

This function will be called by the restart function and the reconfiguration function of the multiprocessor operating system.

Comments:

None.

3.5.2.3.1.2.1.6 FDIR

Multiprocessor Configuration Control

The overall function of the multiprocessor configuration controller shall be to maintain multiprocessor system integrity in the presence of hardware faults. It will do this by detecting faults, identifying the faulty units, and replacing them with spares or gracefully degrading the multiprocessor system if no spares are available. Both hard and transient faults shall be handled by the configuration controller. Spare units will be periodically brought on line. This will include processors, memory units, and clock elements. Spare processor and memory units will be assigned to shadow active processor and memory triads to minimize reconfiguration time. In addition, self test programs will be run continually on active elements to uncover any latent faults.

Function Name: Multiprocessor Fault Detection

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: a cyclic time task, scheduled by operating system
Termination:

Functional Description of Inputs:

Hardware error latches

Functional Description of Outputs:

Faulty unit, error latches cleared

Description of Function:

Fault detection in the multiprocessor will be done by hardware majority voting and subsequent disagreement detection between the majority voter output and each of the three inputs. The disagreements will be stored in hardware registers which for clarity purposes will be referred to as error latches in this document. The error latches will be read and the bus used to read the latches will be examined for corruption since the latches will be read as simplex source non voted data. If the bus reading the latches does not pass appropriate tests it will be marked as faulty. This function will clear the error latches after reading them. The result will be passed to the fault identification function.

Comments:

None.

Function Name: Fault Identification

Requirements Source: CSDL

Modes in Which Function Required:

Fault Processing
Reconfiguration

Initiation and Termination Events:

Initiation: Fault detection function call
Termination: Faulty unit identified

Functional Description of Inputs:

faulty unit, local system configuration tables

Functional Description of Outputs:

Identity of faulty unit

Description of Function:

This function will identify the faulty unit using the output of the fault detection function and the local system configuration tables. It will call the reconfiguration function to determine if the fault is a hard or a transient fault.

Comments:

None.

Function Name: Reconfiguration

Requirements Source: CSDL

Modes in Which Function Required:

Reconfiguration

Initiation and Termination Events:

Initiation: One of the six events occur that are listed below.
Termination: Reconfigured system

Functional Description of Inputs:

The inputs to this include the reason for reconfiguration, the local system configuration tables, the faulty unit.

Functional Description of Outputs:

Updated loeal system configuration tables

Description of Function:

The reconfiguration function must reconfigure the multiprocessor system for the following reasons: to form a new processor triad, to assign shadow processors or shadow memories, to cycle spares, to identify faults for the hard fault analysis function, to deactivate an identified faulty unit.

(1) New Processor Triad Formation

When there are enough spares available to start a new processor triad this function will start up the triad by enabling the three processors on the appropriate buses, assign them an active identifier and activate the initialization and synchronization functions.

(2) Shadow Processor Assignment

This function will assign an available spare processor to 'shadow' an active triad. The word 'shadow' in this context means that the shadow processor will be in tight synchronization with the active triad. It will listen to all commands on the buses and receive all the appropriate interrupts the same as the active members. It will not transmit on the buses. This function will find a processor triad that does not have a shadow, will pass the identifier of the active triad to the spare processor and will activate its synchronization function.

(3) Shadow Memory Assignment

This function will assign an available memory unit to 'shadow' an active memory triad. The word 'shadow' in this context means that the shadow memory will respond to memory write requests, but not memory read requests. This function will assign the appropriate identifier to the shadow unit, enable it on the appropriate buses and refresh its entire memory to agree with the active memory triad.

(4) Spare Cycling

The spare elements in the multiprocessor, processors, memories, buses will be periodically activated. The spare cycling function will determine which spare units should be brought on line and do the appropriate swapping with the active unit.

(5) Diagnostic Reconfiguration

The fault identification function will call this function to reconfigure the system. This function will change the assignment of the faulty unit to determine if the fault is a hard or a transient fault. The transient and hard failures will be handled as follows.

Hard Fault Analysis	If the fault persists for several frames after it has been reassigned, the faulty unit will be identified for deactivation. If the fault does not persist it will analyzed by the transient fault analysis function.
Transient Fault Analysis	This function will be based on the concept of 'fault index.' Each element in the multiprocessor will be assigned a fault index. If there is a transient fault, the faulty unit will be assigned demerits. The fault index will then be recomputed every time a transient fault occurs. If the fault index of a suspect goes above a predetermined threshold, the unit will be deactivated.

(6) Deactivate Faulty Unit

This function has three cases. The failed unit can be a processor, a clock or a memory.

If the failed unit is a processor, the function must replace the failed processor by a suitable shadow or spare. If there are no shadows or spares, this function will retire the faulty processor and make the remaining two good processors spares. The multiprocessor system memory will be updated with the new assignments.

If the faulty unit is a memory, the function must replace the failed memory by a suitable shadow or a spare. The multiprocessor system memory will be updated with the new assignments.

If the faulty unit is a clock, the function shall replace the failed clock with a spare clock. If there is no spare clock available then the faulty clock is disabled. The multiprocessor system memory will be updated with the new assignments.

Comments:

None.

Function Name: Self Tests

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Multiprocessor system startup
Termination:

Functional Description of Inputs:

Local system configuration tables

Functional Description of Outputs:

Identity of faulty unit

Description of Function:

There will be several self tests running when there is available time. The following self test programs may be included: cache RAM memory diagnostic, cache PROM memory test, opcode diagnostic, interrupt diagnostic, and error latch diagnostic.

Comments:

None.

3.5.2.3.1.2.1.7 Management of Dynamic Memory

Function Name: Multiprocessor Dynamic Memory Management

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: application task request, task initialization
Termination: memory allocated

Functional Description of Inputs:

size of allocation needed, pointers to shared memory

Functional Description of Outputs:

pointers to area of shared memory allocated, software exception

Description of Function:

Each task that can be scheduled by the multiprocessor operating system will be assigned an area of shared memory that is used by the operating system for task management. The amount of memory will be dependant on the number of local variables, subprogram calls and other factors that are determined by operating system.

The multiprocessor will have an area of shared memory that will be set aside for the system to allocate and deallocate dynamically for application programs. This function will manage that portion of shared memory by allocating or deallocating the required amount of memory to the calling program. It will also raise a software exception when there is insufficient memory to perform an allocation.

Comments:

None.

Function Name: Multiprocessor Memory Read

Requirements Source: CSDL

Modes in Which Function Required:

Normal

Initiation and Termination Events:

Initiation: on demand

Functional Description of Inputs:

The inputs to this function are the addresses of the local memory locations to be read.

Functional Description of Outputs:

The outputs of this function are the contents of the local memory locations.

Description of Function:

This function responds to a request for the value of the contents of specified local memory locations.

Comments:

None.

Function Name: Multiprocessor Memory Write

Requirements Source: CSDL

Modes in Which Function Required:

Normal

Initiation and Termination Events:

Initiation: on demand

Functional Description of Inputs:

The inputs to this function are the addresses of the local memory locations to be written into and the values to be stored.

Functional Description of Outputs:

The outputs of this function are the altered contents of the local memory locations.

Description of Function:

This function responds to a request to alter the contents of specified local memory locations.

Comments:

None.

3.5.2.3.1.2.1.8 Local I/O interface

Function Name: Local I/O Bus Interface

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On demand
Termination: Local I/O bus transaction completed

Functional Description of Inputs:

The inputs to this function shall include:

- formatted command/data message for local I/O device.

Functional Description of Outputs:

The outputs from this function shall include:

- response from local I/O device
- transaction status.

Description of Function:

This function shall transmit messages to and receive responses from local I/O devices as a service for the local I/O bus manager of the local operating system. One member of the triad processor will transmit the data. Input data will be received by all members of the triad and then written to shared memory to insure source congruency. Upon completion, the status of the transaction shall be returned to the local I/O bus manager.

Comments:

None.

3.5.2.3.1.2.1.9 Local I/O Configuration Management

Function Name: Local I/O Bus Manager

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Computer turn on
Termination: Computer turn off

Functional Description of Inputs:

The inputs to this function shall include:

- current local I/O bus configuration/status tables
- commands/data for I/O device on local I/O bus.

Functional Description of Outputs:

The outputs from this function shall include:

- updated local I/O bus configuration/status tables
- reconfiguration commands to the local I/O bus (if a network)
- data from I/O device on local I/O bus.

Description of Function:

This function shall be responsible for:

- servicing requests for initiating transactions to local I/O devices
- servicing responses from local I/O devices
- resolving contentions for local I/O devices
- detection of faulty I/O devices
- in the case of a local I/O network, detection of node/link faults, network reconfiguration and periodic incorporation of inactive links.

Comments:

None.

3.5.2.3.1.2.1.10 Software Exception Handler

Function Name: Local Operating System Software Exception Handler

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Detection of software exception.
Termination: Return to application task.

Functional Description of Inputs:

The inputs to this function shall include:

- exception identifier
- error log
- task identifier

Functional Description of Outputs:

The outputs from this function shall include:

- updated error log

Description of Function:

Upon a software exception, this function shall log the exception and return to the application specified software error handler. If there is no application specified error handler, the function will perform a standard error fix up.

Comments:

None.

3.5.2.3.1.2.2 System Interfaces

3.5.2.3.1.2.2.1 Synchronization with Global System Time

Function Name: Global Time Synchronization

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Receipt of system time message
Termination: Local time synchronized with system time

Functional Description of Inputs:

The inputs to this function shall include:

- system time message from global computer including:

  system time value

Functional Description of Outputs:

The outputs from this function shall include:

- time drift correction.
- updated local time value.

Description of Function:

This function shall synchronize local processor time with the broadcast system time. Upon receipt of the periodic system time message from the global computer, the enclosed time value will be compensated by known biases and the resultant value used to dedrift the local time.

Comments:

None.

3.5.2.3.1.2.2.2 Local Support for Function Migration

Function Name: Terminate Function Execution

Requirements Source: CSDL,AIPS-83-50

Modes in Which Function Required:

Reconfiguration

Initiation and Termination Events:

Initiation: Command from global computer to terminate function
Termination: Function execution terminated

Functional Description of Inputs:

The inputs to this function shall include:

- message from global computer to terminate specified function.

Functional Description of Outputs:

The outputs from this function shall include:

- message to global computer indicating termination status
- updated local schedule queues.

Description of Function:

Upon command of the global computer, an attempt shall be made to 'gracefully' abort a currently executing function at an application specified check point. If a check point is not reached within a specified time interval, the function shall be terminated immediately. All open data files shall be closed and any locked data shall be unlocked. All tasks relating to the terminated function shall be deleted from the local system queues. A terminated function shall be allowed to execute again at this processing site only when it has been reactivated by the global computer (e.g.as a result of a future function migration). The status of the function termination shall be transmitted to the global computer.

Comments:

None.

Function Name: Transmit Code/Data

Requirements Source: CSDL

Modes in Which Function Required:

Reconfiguration

Initiation and Termination Events:

Initiation: Receipt of code/data transmission request
Termination: Code/data transmitted

Functional Description of Inputs:

The inputs to this function shall include:

- transmission request containing:

  code/data identifier

  destination processing site identifier.

Functional Description of Outputs:

The outputs from this function shall include:

- message to destination processing site containing required code/data.

Description of Function:

Upon receipt of the code/data transmission request, the indicated code/data (including check point data) resident at this processing site shall be transferred to the destination processing site. Transmission may be to mass memory to be retrieved by the destination processing site or,alternately, over the intercomputer network directly to the destination site.

Comments:

None.

Function Name: Migrate Function

Requirements Source: CSDL

Modes in Which Function Required:

Reconfiguration

Initiation and Termination Events:

Initiation: Function migrate command from global computer
Termination: Function migrated and activated

Functional Description of Inputs:

The inputs to this function may include:

- function migrate message from global computer including:

  function identifier

  code/data source identifier

- message from source processing site containing required function code/data

- local schedule queues.

Functional Description of Outputs:

The outputs from this function may include:

- code/data transmission request to source processing site including:

  function code/data identifier

  destination processing site identifier

- message to global computer giving status of function migration

- updated local schedule queues.

Description of Function:

Upon receipt of the function migrate command from the global computer, the indicated function's code/data shall be obtained from mass memory directly or from another processing site utilizing the intercomputer network or mass memory. In either case , the received code/data shall be prepared for execution/use at this processing site (virtual memory page registers initialized,dynamic linker called etc.).

The indicated function shall be scheduled for execution and activated at its initial start address or at a restart check point. The status of the function migration and activation shall be reported to the global computer.

Comments:

None.

3.5.2.3.1.2.2.3 Intercomputer Interface

Function Name: Intercomputer (IC) Network Interface

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On demand
Termination: IC network transaction completed

Functional Description of Inputs:

The inputs to this function shall include:

- formatted message for transmission on IC network.

Functional Description of Outputs:

The outputs from this function shall include:

- formatted message received from IC network
- IC transaction status.

Description of Function:

This function shall transmit and receive messages from the IC network as a service for the IC network functions of the network operating system. Each member of the triad processor will transmit on only one of the redundant layers of the IC network, but will listen to all three layers. In the case of the receiving function, the hardware voters will insure source congruency. The status of input messages shall be determined from hardware error latches and the results passed along to the user IC network function.

Comments: ,

None.

3.5.2.3.1.2.2.4 Network I/O Interface

Function Name: Global I/O Network Interface

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On demand
Termination: Global I/O network transaction completed

Functional Description of Inputs:

The inputs to this function shall include:

- formatted command/data messages for global I/O device

Functional Description of Outputs:

The outputs from this function shall include:

- response from global I/O device.
- transaction status.

Description of Function:

This function shall transmit messages to and receive responses from global I/O devices as a service for the global I/O network functions of the network operating system. One member of the triad processor will transmit the data. Input data will be received by all members of the triad and then written to shared memory to insure source congruency. The status of the transaction shall be returned to the global I/O network function.

Comments:

None.

Function Name: Regional I/O Network Interface

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On demand
Termination: Regional I/O network transaction completed

Functional Description of Inputs:

The inputs to this function shall include:

- formatted command/data message for regional I/O device.

Functional Description of Outputs:

The outputs from this function shall include:

- response from regional I/O device
- transaction status.

Description of Function:

This function shall transmit messages to and receive responses from regional I/O devices as a service for the regional I/O network functions of the network operating system. One member of the triad processor will transmit the data. Input data will be received by all members of the triad and then written to shared memory to insure source congruency. Upon completion, the status of the transaction shall be returned to the regional I/O network function.

Comments:

None.

3.5.2.3.1.2.3 Testing Interfaces

3.5.2.3.1.2.3.1 Normal Test support

Function Name: Normal Test Support

Requirements Source: CSDL

Modes in Which Function Required:

Normal

Initiation and Termination Events:

Initiation: GPC startup
Termination: GPC shutdown

Functional Description of Inputs:

The inputs to this function include faults to be logged, operating system entry identifiers, and watchdog timer update commands.

Functional Description of Outputs:

The outputs of this function is the specified action.

Description of Function:

This function provides the normal testing support for the processor. These functions include: the logging of all hardware and software faults, the maintenence of an operating entry trace, and the maintenence of a watchdog timer.

Comments:

None.

3.5.2.3.1.2.3.2 Special Test support

Function Name: Special Test Support

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: GPC startup
Termination: GPC shutdown

Functional Description of Inputs:

The inputs to this function include processor halt commands, and commands to simulate faults.

Functional Description of Outputs:

The outputs of this function is the specified action.

Description of Function:

This function provides the special testing support for the processor. These functions include halting the processor upon the detection of specified conditions and simulating specified faults for special testing operations.

Comments:

None.

3.5.2.3.2 Network Operating System The Network Operating System (NOS) provides overall management of the AIPS system, and services to support communication and other applications needs. The following section describes the functions which will provide the management and services. The section is structured into four logical parts:

- IC Network Functions - These functions support network services which initialize and maintain a network of reliable computers that communicate with each via the IC network. These functions,in general, reside in the site designated as the Global Computer. See Figure 37 on page 175.

- Global I/O Network Functions - These functions support I/O services provided by the Global I/O Network. These functions,in general, reside in the site designated as the Global Computer. See Figure 38 on page 191.

- Regional I/O Network Functions - These functions support I/O services provided by the Regional I/O Network. These functions,in general, reside in the site designated by the Global Computer as the Regional Manager. See Figure 39 on page 199.

- Mass Memory Functions - These functions support I/O services which support file management on the mass memory device. These functions are, in general, distributed and are available locally. See Figure 40 on page 208.

3.5.2.3.2.1 Intercomputer Communication

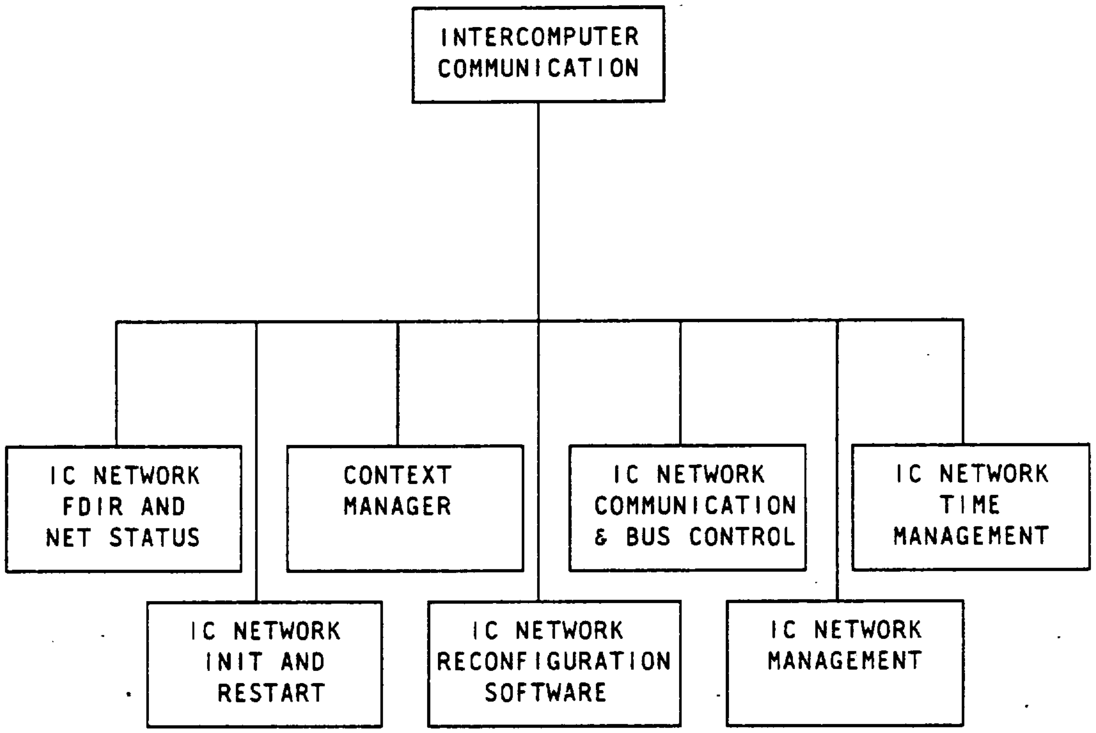
INTERCOMPUTER COMMUNICATION
IC NETWORK FDIR AND NET STATUS
CONTEXT MANAGER
IC NETWORK COMMUNICATION & BUS CONTROL
IC NETWORK TIME MANAGEMENT
IC NETWORK INIT AND RESTART
IC NETWORK RECONFIGURATION SOFTWARE
IC NETWORK MANAGEMENT

Figure 37. Intercomputer Communication Functions

3.5.2.3.2.1.1 IC Network FDIR and Network Status

Function Name: IC Network Fault Identification

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: System Startup
Termination: System Shutdown

Functional Description of Inputs:

The inputs to this process include signals from I/O processes indicating errors in the network.

Functional Description of Outputs:

The output of this process is a database of what errors occurred and when, on transactions in the IC network.

Description of Function:

This process keeps track of errors on the IC network determined by the local I/O processes. These errors are logged in a database for use by management processes to determine when a node or link or processor has failed. The information logged include the type of error, the time at which the error occurred, and the identifiers of the nodes and processors and processes involved.

Comments:

There must be one fault identification process on each subscriber.

Function Name: Subscriber Poll Response

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Upon subscriber receipt of Global Computer poll message.
Termination: Transmission successful.

Functional Description of Inputs:

Inputs to this function consist of site status and any additions to the fault log since the last poll response.

Functional Description of Outputs:

The output of this function is a message to the Global Computer indicating its status and any logged faults.

Description of Function:

This demand function executes in response to the poll message transmitted to each subscriber site. The response to the Global Computer will indicate the site's operational status and any faults that have occurred since the previous poll response. The site status information includes the state and configuration of all elements of the local site that are of concern to the Global Computer functions. This includes software as well as hardware information.

Comments:

This function resides in each processing site and must be available as soon as each site is powered up. For gateways, a similar response shall also be required.

3.5.2.3.2.1.2 IC Network Initialization and Restart

Function Name: IC Network Initialization

Requirements Source: CSDL

Modes in Which Function Required:

Startup
Reconfiguration

Initiation and Termination Events:

Initiation: System Startup or Restart
Termination: Conclusion of Initialization

Functional Description of Inputs:

The inputs to this function will be those needed to determine the state of the network (node, link and processor status). Also needed will be a database of initial process assignments.

Functional Description of Outputs:

The outputs of this process will include:

- The state of the network.
- Functionally, the major output of this process is a correctly configured network, with processes initialized and running.

Description of Function:

This process provides for system initialization for the IC network. The Global Computer must be identified. System time will be obtained. The IC network will be configured and the subscribers polled to determine the state of the system. System time will be broadcast to the subscribers. Finally, the initial function assignments will be given to the appropriate subscribers. Each function is assumed to provide for its own initialization.

Comments:

This is a Global Computer process.

3.5.2.3.2.1.3 Context Manager

Function Name: Interfunction Communication Context Manager

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On Demand
Termination: NA

Functional Description of Inputs:

The inputs to this process will be the inputs to a migratable process, and the outputs from that process.

Functional Description of Outputs:

The outputs from this process will be commands to the IC network to call the appropriate process on a different computer, the inputs to that process, and the outputs from that process.

Description of Function:

In the AIPS application software, there will be no knowledge of the multicomputer nature of the system. Processes at this level will be able to interact with other processes at this level with no knowledge as to where the processes are. The commands used by application level processes to interact with other processes shall be independent of the relative locations of the processes. To this end, there will exist a "context manager" whose task it will be to recognize attempts to interact with external processes and route messages to the context manager on the external processor. The context manager on the receiving end will cause the appropriate action to occur.

Comments:

None.

3.5.2.3.2.1.4 IC Network Reconfiguration Software

Function Name: Reconfigure IC Network

Requirements Source: CSDL

Modes in Which Function Required:

Startup
Reconfiguration

Initiation and Termination Events:

Initiation : On Demand
Termination: When the network is successfully reconfigured

Functional Description of Inputs:

Inputs to this process must specify the recorded status of each component in the global I/O network and the current configuration of the network.

Functional Description of Outputs:

The outputs from this process will include commands to the nodes of the network to reconfigure and the final configuration of the network.

Description of Function:

This process reconfigures the nodes on the IC network such that:

(1) All subscribers will have a communication path to all other subscribers.

(2) Failed nodes are not used.

(3) Failed ports are not used.

(4) "Babbling" subscribers will not be listened to.

This process reconfigures the network based on errors discovered in the components of the network.

Comments:

This process places the network operating system in reconfiguration mode. This is a Global Computer process.

3.5.2.3.2.1.5 IC Network Communication and Bus Control

Function Name: IC Network Communication

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On Demand
Termination: Completion of Service

Functional Description of Inputs:

The inputs to this process include messages and commands from other processes or from the network.

Functional Description of Outputs:

The outputs of this process include:

- Messages routed from the network to a process.
- Messages and commands to the network from a process.
- Error signals to the FDIR process when errors occur.

Description of Function:

This process routes messages from one process on one subscriber to a process on a different subscriber. Protocol standards will be followed when possible.

Comments:

None.

3.5.2.3.2.1.6 IC Network Management

Function Name: IC Network Manager

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Global Computer Startup
Termination: Global Computer Turnoff

Functional Description of Inputs:

The inputs to this process include:

- The state information from the nodes, links and processors.

Functional Description of Outputs:

The outputs of this process include:

- Commands to reconfigure the network.
- Signal to the function migration process to reconfigure.

Description of Function:

This process keeps the system operational. If hardware errors occur, this process will reconfigure the network (if necessary) or invoke the function migration process (if necessary). At periodic intervals, this process will poll the subscribers and nodes. Error information will be obtained from these sources. Based on this information, this process will decide whether the network is reliable; if not, it will reconfigure the nodes (if the failure is in the network) or it will call the function migration process to move functions (if a processor has become unreliable).

Comments:

This process is dependent on the number and type of nodes, along with the protocols used to command a nodal switch. This is a Global Computer process.

Function Name: Initial Program Load

Requirements Source: CSDL

Modes in Which Function Required:

Startup

Initiation and Termination Events:

Initiation: Power up interrupt
Termination: Upon completion

Functional Description of Inputs:

Vectored interrupt to bootstrap loader

Functional Description of Outputs:

This function has no output

Description of Function:

Upon interrupt, each network subscriber shall bootstrap its operating system and any required communication software. It will connect itself to local (if any), and IC networks then idle. If a site is designated as the Global Computer, it shall assume control of the network functions if it can. This entails:

(1) initiating a periodic poll of subscribers

(2) establishing a viable IC network configuration

(3) establishing the global I/O network

(4) assigning regional network manager(s)

(5) setting a value for system time

Sites designated an alternate Global Computer shall, if the designated Global fails during the IPL, resolve any conflicts such that one alternate assumes the Global Computer function. The number of alternates available shall be at least one.

Comments:

None.

Function Name: Subscriber Applications Program Load

Requirements Source: CSDL

Modes in Which Function Required:

Startup
Reconfiguration

Initiation and Termination Events:

Initiation: initial configuration or reconfiguration
Termination: applications program load complete

Functional Description of Inputs:

Initial and alternate processing site load (function) assignments.

Functional Description of Outputs:

Load message to individual subscriber with load information.

Description of Function:

Once an initial physical configuration is established, the Global Computer shall invoke this function to initiate the applications program load configuration. This configuration shall accommodate one of the a priori load assignments. Each subscriber shall be commanded to begin its APL function referencing information in the load message.

When a reconfiguration is required, this function shall be activated in a similar fashion to redistribute functional capabilities.

Upon completion of each site's load, the site shall notify the Global Computer of its completion.

Comments:

This function resides in the Global Computer. The success or failure of this load process is dependent on the availability of a full or alternate complement of hardware resources.

Function Name: Periodic Subscriber Poll

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Upon completion of Global Computer bootstrap load
Termination: always active

Functional Description of Inputs:

Subscriber response to poll message

Functional Description of Outputs:

Poll message transmitted over IC net

Description of Function:

The Global Computer site shall periodically poll all other sites. The poll message shall contain, at a minimum, a request to respond to the poll to each site. The motivation for this poll is twofold, first to assure that communication on the IC net is working, and second to query those sites that have been taken out of the operating configuration to determine if reinstatement actions should be taken. If either the existing network fails or if a processor is to be reinstated, the function migration or network reconfiguration task shall be invoked.

Comments:

This function resides in the Global Computer.

Function Name: Function Migration

Requirements Source: CSDL

Modes in Which Function Required:

Reconfiguration
Normal

Initiation and Termination Events:

Initiation: By the following:

(1) Operator initiated action.

(2) Pre-planned mission phase event or time.

(3) Pre-planned function migration in response to a fault.

(4) Reinstatement of a processing site or device after repair.

Termination: Site response of migration status

Functional Description of Inputs:

Event or command containing the stimulus (see initiation events).

Functional Description of Outputs:

Command and message sequence to sites affected by the function migration, updated site/function tables.

Description of Function:

Each GPC in the AIPS shall be assigned to perform a fixed set of functions. Several alternative assignments, corresponding to different site/function configurations, shall be possible. These alternative configurations shall be resident in the non volatile program store, and be accessible to all of the pre designated sites through the mass memory or some other data transfer medium. The control of this function shall reside in the site designated as the Global Computer. Information necessary for interprocessor and intertask communication shall be maintained and made available to subscriber sites each time a configuration assignment is altered.

This function shall be activated during normal operating mode only in response to the stimuli listed above, to exercise a controlled function migration. Controlled implies that an a priori decision has been made that defines what functions can migrate to where.

In case (1), operator inputs shall be checked for correctness and adherence to predefined allowable configurations prior to changing the system configuration. In case (2), an overlay or total memory reload may be automatically initiated. In case (3), the global manager shall either (1) command the "from" processor to transfer tasks and data directly to the "to" processor via the IC network (e.g., large block transfer), or (2) command the "to" processor to begin executing a task already resident in its memory, or (3) command the "to" processor to retrieve the new tasks and data from mass memory and begin executing the new tasks.

In any of the cases listed, the global manager shall determine if a reconfiguration is possible (i.e., there is a capable alternate site available with the required redundancy level to assume the new role). If the reconfiguration is not possible, an exception shall be raised alternative action shall be taken.

When global functions migrate they shall move to a designated alternate. If reconfiguration is possible, the alternate shall be commanded to initiate its local function migration tasks and assume the Global Computer role when it can be gracefully done.

Comments:

In all of the above cases, there shall be only one function migration task in process at any time. This function resides in the Global Computer.

3.5.2.3.2.1.7 Network Time Management

Function Name: Broadcast System Time

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Global Computer startup
Termination: Global Computer shut down

Functional Description of Inputs:

Time value read from the source of system time and the base offset value for system time.

Functional Description of Outputs:

Broadcast transmission of the System Time on the IC bus.

Description of Function:

This global computer function periodically reads the source for system time, adds the time base offset value to arrive at System Time, and then transmits it to all functioning subscribers on the IC network.

Comments:

None.

Function Name: Set System Time

Requirements Source: CSDL

Modes in Which Function Required:

Startup
Normal

Initiation and Termination Events:

Initiation: On demand

Functional Description of Inputs:

Desired value of System Time.

Functional Description of Outputs:

The output of this function is a new setting of System Time.

Description of Function:

This global computer function alters the value of System Time in response to an operator request or upon system initialization.

Comments:

None.

Function Name: System Time Source FDIR

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: Global Computer startup
Termination: Global Computer shut down

Functional Description of Inputs:

GPC local clock dedrift values and an indication of the source of system time.

Functional Description of Outputs:

Selection of the source of system time.

Description of Function:

This function selects the source for system time.

Comments:

None.

3.5.2.3.2.2 Global I/O Communication

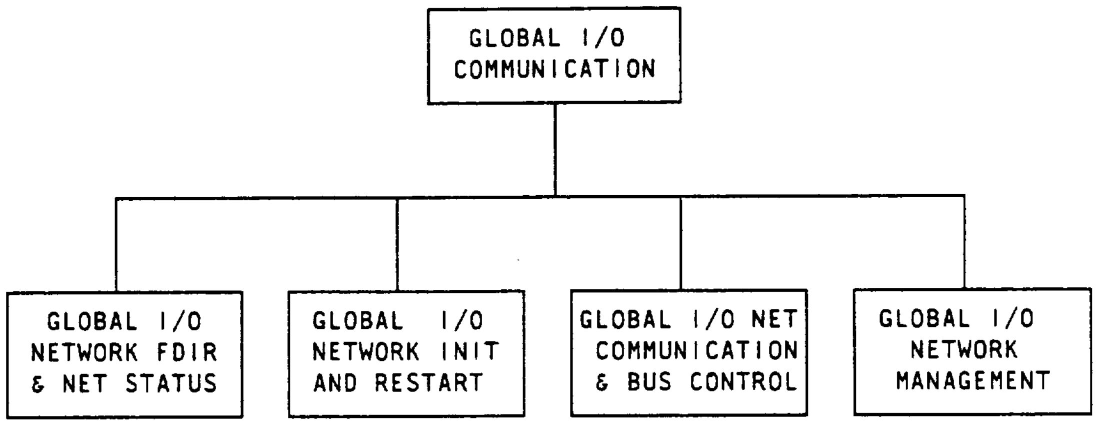
GLOBAL I/O COMMUNICATION
GLOBAL I/O NETWORK FDIR & NET STATUS
GLOBAL I/O NETWORK INIT AND RESTART
GLOBAL I/O NET COMMUNICATION & BUS CONTROL
GLOBAL I/O NETWORK MANAGEMENT

Figure 38. Global I/O Communication Functions

3.5.2.3.2.2.1 Global I/O Network FDIR and Network Status

Function Name: Signal Message Failure

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : On demand when a transmission fails
Termination:

Functional Description of Inputs:

None.

Functional Description of Outputs:

The outputs from this function must specify the source computer of the transmission and the destination device of the transmission.

Description of Function:

This function sends a message to the Global Computer that a transmission was unsuccessful on the global I/O network.

Comments:

This function should be operable from each computer on the global I/O network.

3.5.2.3.2.2.2 Global I/O Network Initialization and Restart

Function Name: Initialize Global I/O Device

Requirements Source: CSDL

Modes in Which Function Required:

Startup

Initiation and Termination Events:

Initiation : On demand when a device changes status to "active"
Termination:

Functional Description of Inputs:

None

Functional Description of Outputs:

None

Description of Function:

This function performs any initializations required by a device.

Comments:

This function should be operable from any Global Computer on the global I/O network.

3.5.2.3.2.2.3 Global I/O Network Communication and Bus Control

Function Name: Transmit to Global I/O Device

Requirements Source:

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : On demand when requested by an application or system function
Termination:

Functional Description of Inputs:

The inputs to this function must specify the data to be sent to the device and the device to which it is to be sent.

Functional Description of Outputs:

The outputs of this function must specify any data received as a response from the device and whether or not the transmission was successful.

Description of Function:

This function provides the service of transmitting commands and/or data to a device on the global I/O network and receiving a response from the device.

Comments:

This function should be operable from each computer on the global I/O network.

3.5.2.3.2.2.4 Global I/O Network Management

Function Name: Manage Contentions for Global I/O Device

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : When a global I/O device changes status to connected
Termination: When the global I/O device changes status to isolated

Functional Description of Inputs:

The inputs to this function must specify an I/O device on the global I/O network.

Functional Description of Outputs:

The outputs from this function must specify when a process may communicate with the device through the global I/O network.

Description of Function:

This function decides which of two or more contending processes may communicate with a device on the global I/O network when an uninterrupted sequence of multiple transmissions is required to complete a communication. During such a communication, all other transmissions must be blocked until the communication is completed.

Comments:

The need for this function is dependent upon the device and the nature of the communications with the device. Contentions between single transmission communications are decided by the contention hardware on the network.

Function Name: Reconfigure Global I/O Network

Requirements Source: CSDL

Modes in Which Function Required:

Startup
Reconfiguration

Initiation and Termination Events:

Initiation : Periodically to exercise "inactive" links, on demand when a fault is detected, and on demand when a node changes status to "active".
Termination: When the network is successfully reconfigured

Functional Description of Inputs:

Inputs to this function must specify the recorded status of each component in the global I/O network, the current configuration of the network, and information concerning any new faults in the system.

Functional Description of Outputs:

The outputs from this function must specify any changes in the status of components in the global I/O network or in the current configuration of the network.

Description of Function:

This function reconfigures the nodes on the global I/O network such that:

(1) there is a path from the Global Computer to every operating node in the network.

(2) links that have had the status "inactive" for a period of time are changed to "active"

This function isolates, identifies, and recovers from errors discovered in the components of the network.

Comments:

This function should be operable from any Global Computer on the global I/O network. It places the network operating system in reconfiguration mode.

Function Name: Periodically Check Status of Global I/O Network Nodes

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : Periodically beginning at startup
Termination: None

Functional Description of Inputs:

None

Functional Description of Outputs:

The outputs from this function must specify the status and configuration of the nodes, specifically:

- The configuration of the ports to the node, i.e., which ports are allowed to transmit messages
- The number of bit errors recorded for each port on the node since the last status check
- The status of each port of each node

Description of Function:

This function commands the nodes to report status and error log information to the Global Computer.

Comments:

This function should be operable from any Global Computer on the global I/O network.

Function Name: Periodically Check Idle Global I/O Device

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : Periodically beginning at startup
Termination: None

Functional Description of Inputs:

The inputs to this function must specify a global I/O device.

Functional Description of Outputs:

The outputs from this function must specify whether or not the transmission was completed successfully.

Description of Function:

This function periodically sends a transmission to a device that has been idle for a period of time to determine if the device and path to the device are still functioning.

Comments:

None.

3.5.2.3.2.3 Regional I/O Communication

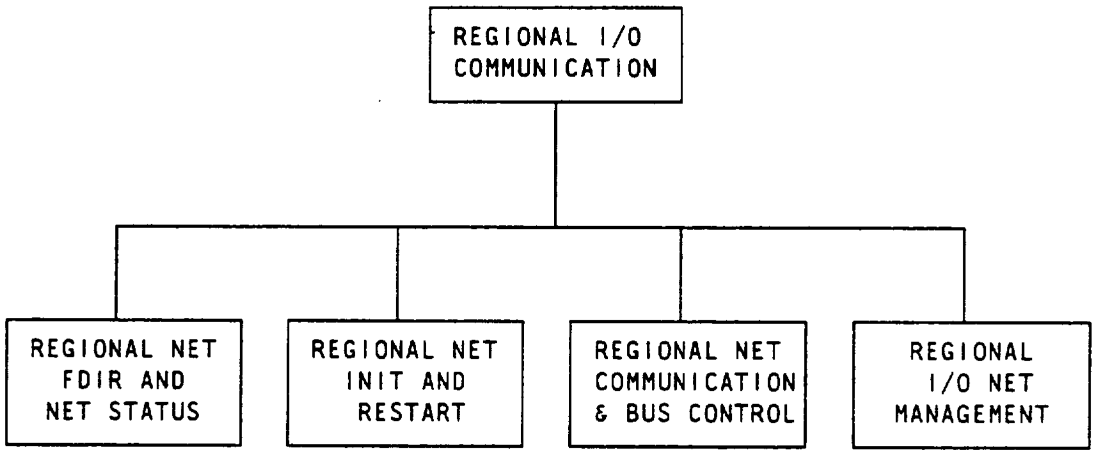
REGIONAL I/O COMMUNICATION
REGIONAL NET FDIR AND NET STATUS
REGIONAL NET INIT AND RESTART
REGIONAL NET COMMUNICATION & BUS CONTROL
REGIONAL I/O NET MANAGEMENT

Figure 39. Regional I/O Communication Functions

3.5.2.3.2.3.1 Regional Network FDIR and Network Status

Function Name: Signal Message Failure

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : Transmission failure on a regional I/O network
Termination:

Functional Description of Inputs:

Functional Description of Outputs:

Identification of source and destination device for the transmission.

Description of Function:

Sends a message to the regional manager indicating that a transmission was unsuccessful on the regional I/O network.

Comments:

None.

3.5.2.3.2.3.2 Regional Network Initialization and Restart

Function Name: Initialize Regional Network

Requirements Source: CSDL

Modes in Which Function Required:

Startup
Reconfiguration

Initiation and Termination Events:

Initiation : On demand
Termination:

Functional Description of Inputs:

Command from the global manager to a selected GPC to initialize a regional I/O network.

Functional Description of Outputs:

Tables indicating configuration and status of devices, computers, and busses comprising the regional network.

Description of Function:

Initializes a regional I/O network. Initializes each I/O device on the regional network.

Comments:

GPC selected by the global manager to initialize a regional network becomes the manager of that network.

Function Name: Initialize Regional I/O Device

Requirements Source: CSDL

Modes in Which Function Required:

Startup

Initiation and Termination Events:

Initiation : On demand when a device changes status to "active"
Termination:

Functional Description of Inputs:

Identity of regional I/O device.

Functional Description of Outputs:

None

Description of Function:

Function performs any initializations required by a regional I/O device.

Comments:

None.

3.5.2.3.2.3.3 Regional Network Communication and Bus Control

Function Name: Transmit to Regional I/O Device

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : Request by an application or system function
Termination:

Functional Description of Inputs:

Identification of data to be sent and destination device.

Functional Description of Outputs:

Indication of whether the transmission was successful.

Description of Function:

Provides for the transmitting of command and/or data to a device on the regional I/O network. Provides for the response of a device to a transmission.

Comments:

None.

3.5.2.3.2.3.4 Regional I/O Network Management

Function Name: Manage Contentions for Regional I/O Device

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : Status of regional I/O device changes to connected
Termination: Status of regional I/O device changes to isolated

Functional Description of Inputs:

Name of an I/O device on the regional network.

Functional Description of Outputs:

Permission for a process to communicate with a device through the regional network.

Description of Function:

Function decides which of two or more contending processes may communicate with a device on the regional I/O network when an uninterrupted sequence of multiple transmissions is required. During such a communication, all other transmissions are blocked until the communication is completed.

Comments:

The need for this function depends upon the device and the nature of the communications with the device. Contentions between single transmission communications are decided by the contention hardware on the network.

Function Name: Reconfigure Regional I/O Network

Requirements Source: CSDL

Modes in Which Function Required:

Startup
Reconfiguration

Initiation and Termination Events:

Initiation : Periodically to exercise "inactive" links, on demand when a fault is detected, and on demand when a node changes status to "active".
Termination: Upon successful reconfiguration of the network

Functional Description of Inputs:

Recorded status of each component in the regional I/O network, the current configuration of the network, and information concerning any new faults in the system.

Functional Description of Outputs:

Identification of any changes in the status of components in the regional I/O network or in the current configuration of the network.

Description of Function:

Function reconfigures the nodes on the regional I/O network such that:

(1) there is a path from the regional manager to every operating node in the network.

(2) links that have had the status "inactive" for an application dependent period of time are changed to "active"

Function isolates, identifies, and recovers from errors discovered in the components of the network.

Comments:

None.

Function Name: Periodically Check Status of Regional I/O Network Nodes

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : Periodically every application dependent time interval beginning at startup
Termination: None

Functional Description of Inputs:

None

Functional Description of Outputs:

Status and configuration of the nodes, specifically:

- The configuration of the ports to the node, i.e., which ports are allowed to transmit messages
- The number of bit errors recorded for each port on the node since the last status check
- The status of each port of each node

Description of Function:

Function commands the nodes to report status and error log information to the regional manager.

Comments:

None.

Function Name: Periodically Check Idle Regional I/O Device

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation : Periodically every application dependent time interval beginning at startup
Termination: None

Functional Description of Inputs:

Name of a regional I/O device.

Functional Description of Outputs:

Indication of whether the transmission was completed successfully.

Description of Function:

Function periodically sends a transmission to a device that has been idle for an application dependent period of time to determine if the device and path to the device are still functioning.

Comments:

None.

3.5.2.3.2.4 Mass Memory Communication

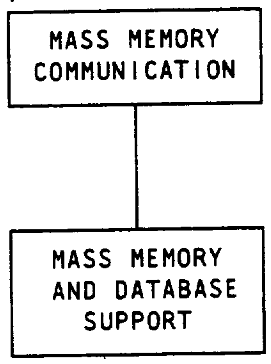

Figure 40. Mass Memory Communication Function

3.5.2.3.2.4.1 Mass Memory and Database Support

Function Name: File Management

Requirements Source: CSDL

Modes in Which Function Required:

All

Initiation and Termination Events:

Initiation: On Demand
Termination:

Functional Description of Inputs:

Various

Functional Description of Outputs:

Various

Description of Function:

The file management function shall support the following subfunctions:

- Initialize File System

  Initializes the global file system.

- Create File

  File with desired name is created.

- Open File

  Function prepares the files to be accessed by the requester and "locks" the files thereby preventing other processes from accessing the files.

- Cancel Open Request

  Cancels previous open file request.

- Read File

  Requested file transmitted to user.

- Write File

  Designated file is updated.

- Close File

The designated file is unlocked thereby freeing it for access by others.

- Delete File

  All references to designated file in the file system directory are deleted.

- Directory Functions

  Functions are used for querying and updating file system directory.

- FDIR Functions

  Functions provide fault detection, isolation, and recovery mechanisms for the file system.

Comments:

None.

Function Name: Mass Memory Read

Requirements Source: CSDL

Modes in Which Function Required:

Normal

Initiation and Termination Events:

Initiation: on demand

Functional Description of Inputs:

The inputs to this function are the addresses of the mass memory locations to be read.

Functional Description of Outputs:

The outputs of this function are the contents of the mass memory locations.

Description of Function:

This function responds to a request for the value of the contents of specified mass memory locations.

Comments:

None.

Function Name: Mass Memory Write

Requirements Source: CSDL

Modes in Which Function Required:

Normal

Initiation and Termination Events:

Initiation: on demand

Functional Description of Inputs:

The inputs to this function are the addresses of the mass memory locations to be written into and the values to be stored.

Functional Description of Outputs:

The outputs of this function are the altered contents of the mass memory locations.

Description of Function:

This function responds to a request to alter the contents of specified mass memory locations.

Comments:

None.

4.0 QUALITY ASSURANCE

4.1 General Provisions shall be made for assuring the quality of the POC system by performing a variety of inspection activities which assess the compliance of the POC system with the requirements of this specification.

4.1.1 Responsibility For Tests Quality assurance activities shall be conducted at CSDL by CSDL personnel. However, the NASA reserves the right to witness such activity.

4.2 Quality Conformance The following methods of inspection are to be used for quality assurance of the POC system.

(1) Examination (E).This form of inspection will consist of investigation, without the use of special laboratory appliances or procedures, to determine compliance with requirements. Examination is generally nondestructive and includes (but is not limited to) visual inspection, simple physical manipulation, gauging, and measurement.

(2) Demonstration (D). This form of inspection is limited to readily observable functional operation to determine compliance with requirements. Demonstrations do not require the use of special equipment or sophisticated instrumentation.

(3) Test (T). This form of inspection employs technical means including (but not limited to) the evaluation of functional characteristics by use of special equipment or instrumentation, simulation techniques, and the application of established principles and procedures to determine compliance with requirements. The analysis of data derived from test is an integral part of this type of inspection.

(4) Analysis (A). Analysis is an element of inspection, taking the form of the processing of accumulated results and conclusions, intended to prove that verification of requirements has been accomplished. The analytical results may be comprised of a compilation or interpretation of existing information or derived from lower level examinations, tests, demonstrations, or analyses.

Table 2 on page 214 indicates those methods to be applied for confirmation of the requirements stated in section 3 of this specification.

Table 2. Quality Assurance Methods vs Requirements

Section 3 Paragraph		E	D	T	A
3.1	System definition				
3.1.1	Introduction				
3.1.2	AIPS concept		x		x
3.1.3	System Services		x	x	x
3.1.4	Proof-of-Concept System		x	x	x
3.1.5	AIPS POC System Operating Environment		x	x	x
3.2	Characteristics				
3.2.1	Performance goals		x	x	x
3.2.2	Physical characteristics	x			
3.2.3	Design objectives		x	x	x
3.2.4	Testability		x	x	x
3.2.5	Environmental conditions	x			
3.3	Design and construction				
3.3.1	Parts, materials & processes	x			
3.3.2	Electromagnetic radiation	x			
3.3.3	Workmanship	x			
3.3.4	Computer programming standards	x			
3.4	Documentation	x			
3.5	Functional area characteristics				
3.5.1	System hardware		x	x	x
3.5.2	System software	x	x	x	x

4.3 Evaluation Evaluation testing will be accomplished after completion of development inspections. Development inspections include testing and other activities accomplished to assure that the various AIPS requirements are satisfied. Evaluation tests will exercise the AIPS in stress environments to explore the achieved performance and characteristics of the system. This testing will be conducted on the Integration and Evaluation Test Facility.

5.0 PREPARATION FOR DELIVERY

5.1 Shipments All shipments shall be accomplished in accordance with CSDL's established shipping methods and procedures and shall be properly documented by the CSDL's usual shipping forms, memos, etc. The NASA shall not directly or indirectly be charged any sum for insurance coverage on items shipped. Shipping documents will indicate the applicable Prime Contract as being NAS9-16023, and the applicable Task Order as being Task Order No. 84-18. Information copies of all letters of transmittal of reports, shipping documents or other relevant delivery information will be forwarded to the NASA/JSC Contracts Administration Office, Attention: A. M. Cornelius, BC28.

6.0 NOTES

6.1 Laning Poll

6.1.1 Overview The Laning poll is a method by which subscribers compete for access to a bus-like resource. Subscribers wishing access to the medium detect its availability by completion of a transaction or a period of nonuse and perform a cooperative sequence which results in control of the medium being granted to one contender.

- The bus must act as a logic 'or' to signals presented to it; that is, if any subscriber outputs a "1", all subscribers will receive a "1".
- Each subscriber must have a unique number associated with it. All other things being equal, the subscriber having the highest number will prevail.
- Dynamic priority may be implemented. The results will be that the participant having the highest priority (as determined by that participant) will gain access. If two or more contenders have the same priority the higher numbered participant will prevail.

6.1.2 An Implementation A serial implementation of the Laning poll proceeds in the following sequence:

(1) A subscriber desiring access transmits a "1" on the bus to signifying the beginning of a competition. Others may do this simultaneously, of course. At this point, all subscribers are aware that a poll sequence has begun and may participate if they desire access; if they do not want access at this time, they must not join the sequence in progress.

(2) Contenders next transmit the most significant bit of their priority. If a 'zero' appears on the bus, all contenders continue. If a 'one' appears of the bus, contenders transmitting a 'one' continue, while contenders transmitting a 'zero' have been eliminated and no longer participate in this sequence.

(3) The remaining bits of priority are transmitted, then the bits of subscriber number. If during any bit interval a 'one' appears on the bus while a contender is transmitting 'zero' that contender ceases participation. The result is that a number has been transmitted to all subscribers indicating the priority and number of the successful subscriber, which now transmits data.

The number of bit intervals required to conduct the poll is:

$$B = 1 + \log_2 P + \log_2 S$$

where P = number of possible priority levels
S = number of possible subscribers

6.1.3 Timing Considerations. For proper operation, all subscribers must operate simultaneously on bits of the same significance. This implies that a bit interval during a poll must be of a duration of at least one roundtrip delay on the medium.

6.2 Encoded Mass Memory A triplicated memory provides adequate fault masking capability. However, it is too costly to triplicate a 10 megaword memory. A very attractive alternative for such a large memory array is to use coded redundancy. Coded redundancy, when used appropriately, can provide the same level of fault tolerance as replicated memory. And it is much more efficient in terms of hardware than replicated memory for large memory arrays.

A candidate coded redundancy scheme for the mass memory is shown in Figure 41 on page 217.

A triplex mass memory multiplex bus connects General Purpose Computers to a Mass Memory Unit (MMU). Within the MMU there are three Encoder/Decoder Units (EDUs). Each EDU interfaces with one of the three Mass Memory Buses. An EDU receives 16-bit data words and the address and command words for memory access operations from a processor. There are also N memory modules which actually store the data. Each of the memory modules interfaces with each of the EDUs. The operation of the MMU is as follows.

For memory write operations, data received from a processor is encoded by the EDU and appropriate field of the coded word is sent to each memory module. Each memory module receives three copies of a part of the coded data, votes on it, and stores the voted result.

For memory read operations, each memory module responds to the read request with a part of the coded word to each of the three EDUs.. The EDUs assemble memory responses into the coded word, decode it into the 16-bit data word, and transmit it to processors on the Mass Memory Bus. Each processor then listens to the three buses and votes on the three copies of the data word.

An MMU with six memory modules is shown in Figure 41 on page 217.

The encoding/decoding of such a memory is as follows. For memory write operations, each EDU converts the 16-bit data word into a 24-bit code word using Hamming Code. The 24-bit code word is then split into six 4-bit fields. Each of these six 4-bit fields is stored in an independent memory module. For an 8 megaword mass memory, each memory module will be $8X2^{20}$ by 4. The 23-bit memory address is passed to the memory modules unchanged.

For memory read operations, each memory module responds to the EDUs with a 4-bit field of the coded data word. The six 4-bit fields are received by each EDU, assembled into a 24-bit code word, and then decoded to retrieve the 16-bit data word.

This coded redundancy scheme allows one to tolerate a complete failure of any one of the memory modules. One can also tolerate multiple bit failures in different memory modules as long as they are not in the same word. And, of course, failures of any one of the three Encoder/Decoder Units is also masked by the voters in the processor and the memory modules.

Thus, single fault masking capability can be achieved with an overhead of only 50 per cent over simplex in a coded redundancy scheme versus 200 per cent for a triplicated memory. The coded memory requires 12 megawords of

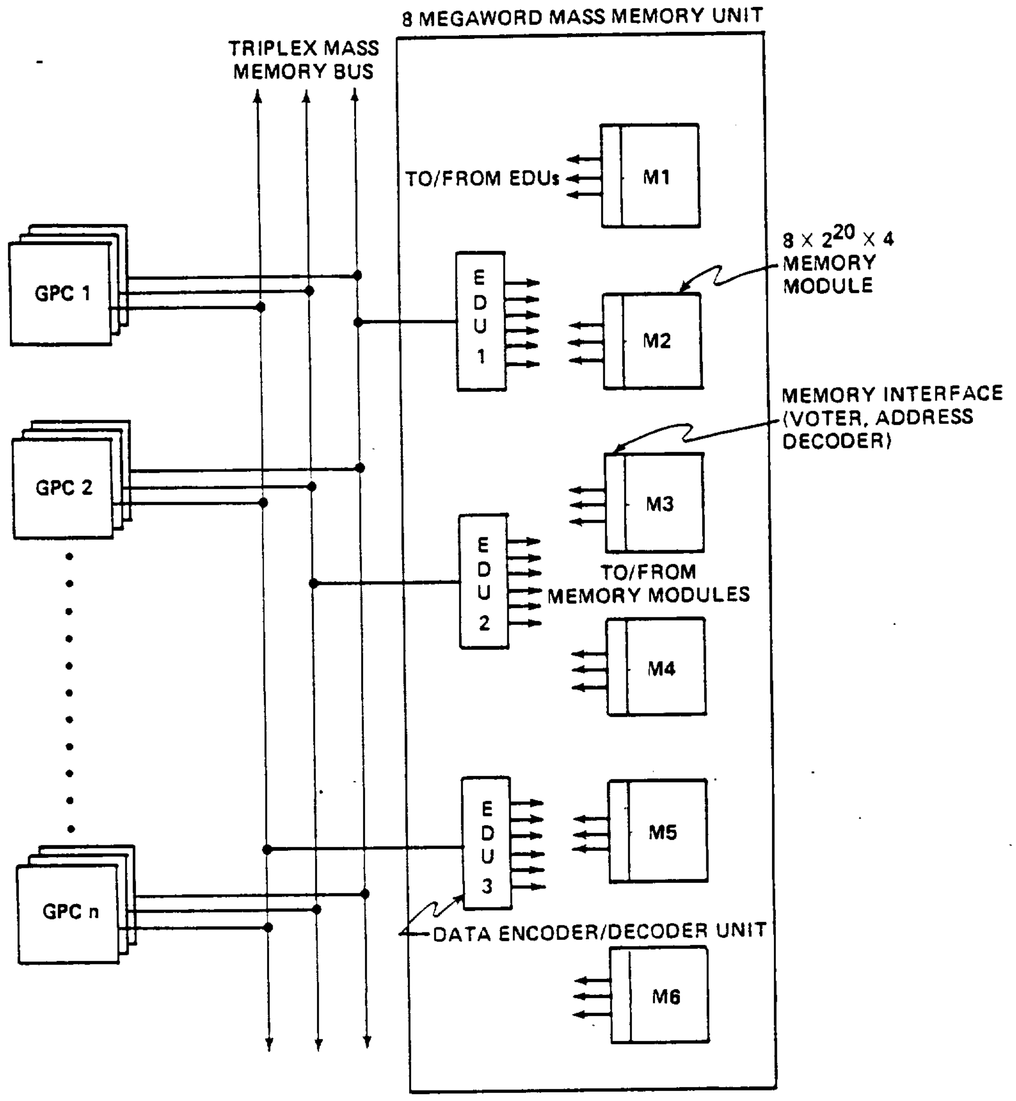
8 MEGAWORD MASS MEMORY UNIT
TRIPLEX MASS MEMORY BUS
TO/FROM EDUs
M1
$8 \times 2^{20} \times 4$ MEMORY MODULE
GPC 1
EDU 1
M2
MEMORY INTERFACE (VOTER, ADDRESS DECODER)
GPC 2
M3
EDU 2
TO/FROM MEMORY MODULES
M4
EDU 3
M5
GPC n
DATA ENCODER/DECODER UNIT
M6

Figure 41. Mass Memory Architecture

physical memory while triplicated memory requires 24 megawords to implement 8 megawords of logical memory.

6.3 Simplex Source Congruency Simplex source congruency is defined as congruent or identical distribution of data from a simplex source to a redundant system. This simplex source of data may be within or without the system. It is important that all redundant copies of hardware receive congruent values of data originating in the simplex source. This is a necessary condition in fault tolerant systems that rely on exact replication and comparison of computational streams. The actual means of achieving exact replication are not relevant. Redundant computers, whether tightly synchronized or only frame synchronized, must perform source congruency operation on all simplex data. In the absence of source congruency even a triply redundant system can suffer single point system failure.

Figure 42 on page 219 illustrates the concept of simplex source congruency for a triplex system.

Consider a radar altimeter that is connected to one channel of a triplex computer as shown in Figure 42. Typically, at the beginning of a control law computation, channel A would read the altimeter as an input. It then sends this value to channels B and C, as illustrated by step I in the figure. In the absence of any faults, all channels will have an identical value of the radar altitude of the aircraft. After all sensors have been read and distributed to all channels, control law computation will be performed and all three processors should produce identical actuator commands. Now, assume that there is a fault in channel A's transmitter such that the values transmitted to B and C are not the same. Let us assume that C receives the correct value but the value received by B is different. In this case, A and C would produce identical results. However, B's answer should be different. Since channel outputs are normally compared to check for faults in the computational core, channel B would be flagged as having failed. That is, the fault diagnosis in this case would be incorrect. In fact, the results can be even more catastrophic if a different value is transmitted to each channel. All three channels can not only produce divergent actuator commands but can also be totally desynchronized from each other, depending upon the nature of the input value.

One way to avoid this single point failure is to exchange and vote upon the value received by each channel. This is shown as step II in Figure 42. In step II, all channels retransmit the word to each other. Each channel votes upon the three values and the majority output of the voter is used as the congruent value of the simplex input. Now, if channel A does not transmit the same value to all in step I, or if A, B, or C do not transmit the same value in step II, even then, the voted value will be identical in all the channels provided that there is only a single fault. This implies that steps I and II should not be affected simultaneously by a correlated fault. Therefore, it is necessary to provide a fault containment region around each transmitter. It can be shown that it is necessary to have four fault containment regions to tolerate all single faults. Three fault containment regions such as those in most triplex systems are not sufficient.

It should be noted here that performing source congruency operations on simplex data only assures that all redundant processors have a congruent value. It does not assure that the congruent value is correct. Other means must be employed to ascertain the validity of the value. Typically, for critical values the source, such as a sensor, is replicated. Each of the redundant sensors is read in by a processor and the congruent value of each sensor distributed to all processors. All processors can then perform appropriate sensor fault detection and isolation algorithm to arrive at a correct and congruent sensor value.

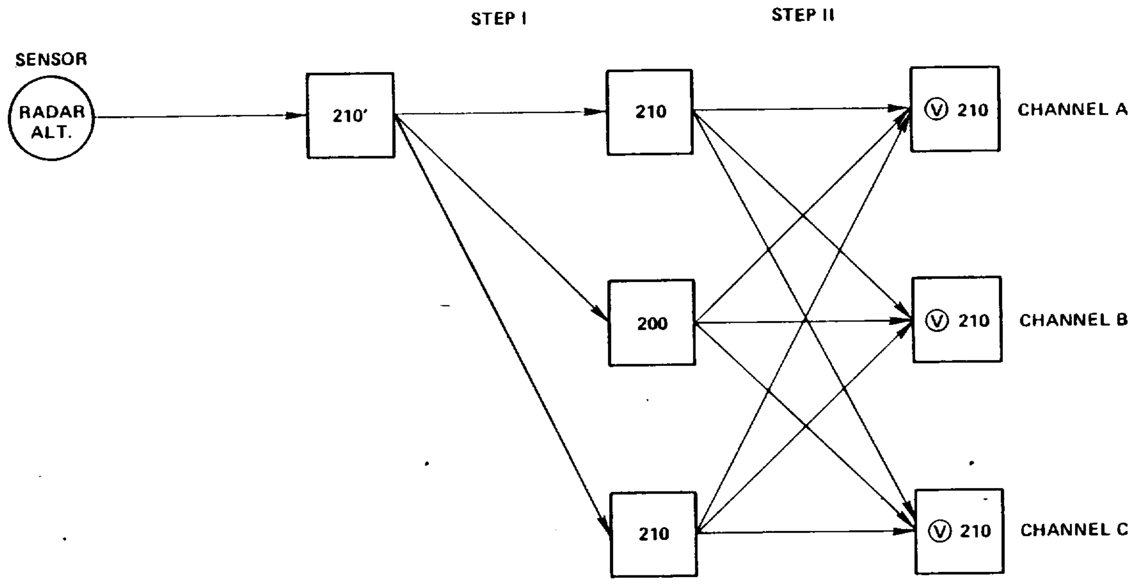

Figure 42. Simplex Source Congruency: An Example